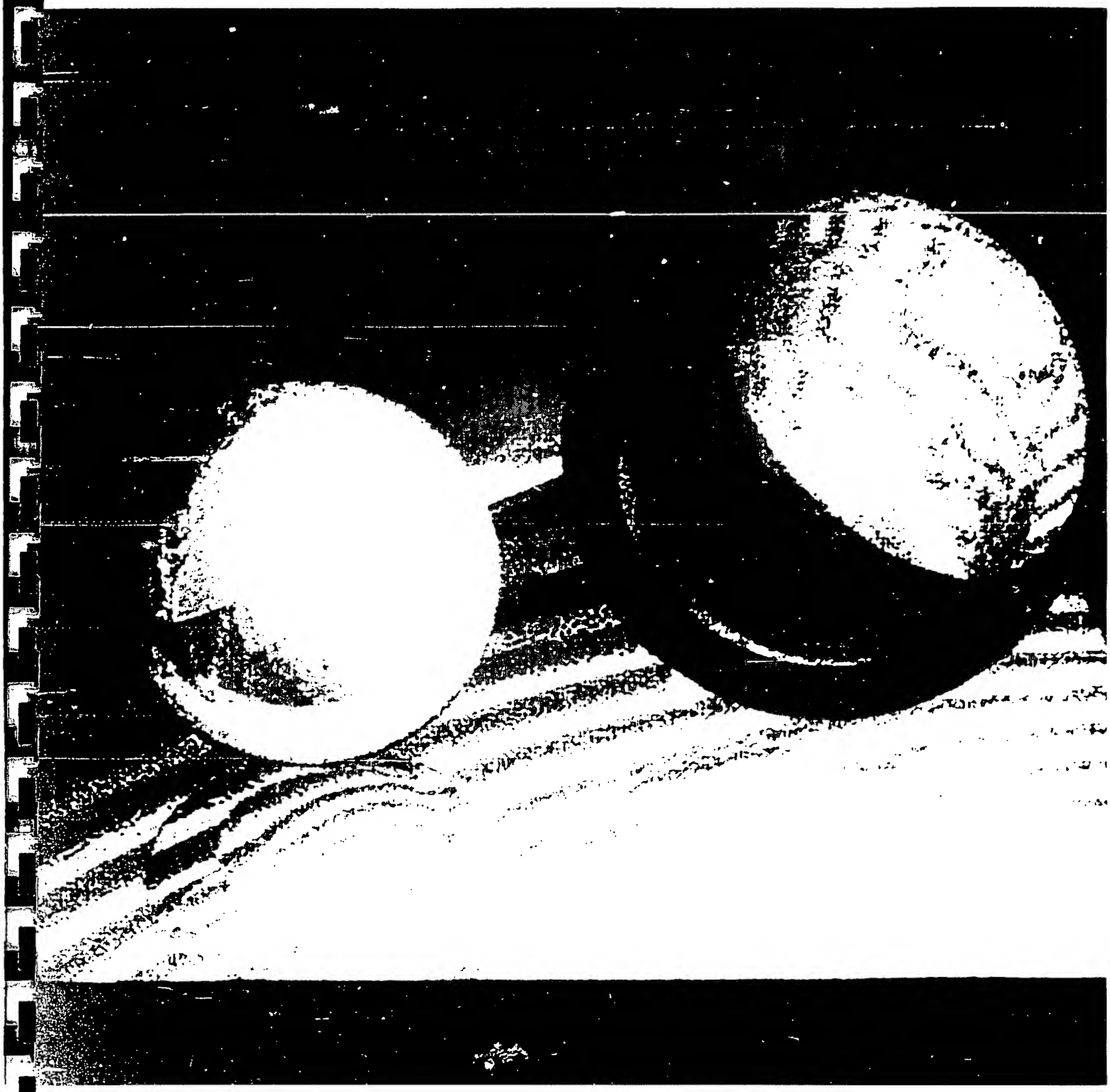
Appendix to Reexamination Request for Patent 5,806,063

1. Shaughnessy U.S. Patent 5,630,118
2. Ohms, "Computer Processing of Dates Outside the Twentieth Century", *IBM Systems Journal*, v.25, #2, 1986, pp 244 et seq.
3. Japanese Published Application of Hazama, JP-5-27947, February 5, 1993
4. Japanese Published Application of Saka, JP-6-103133, April 15, 1994
5. Figure 2, as filed
6. Amendments to claims 1 and 11
7. Exhibit A
8. Certificate of Correction
9. Claim Tables

IBM

Systems Journal

Vol. 25, No. 2, 1986



IBM
Systems Journal
Vol 25 No 2, 1986

Computer processing of dates outside the twentieth century

by B. G. Ohms

This paper presents practical solutions to problems envisioned in extending computer processing of dates beyond the twentieth century. Many data processing managers are concerned with processing cross-century dates, and in doing so using existing systems, with a minimum of disruption to normal operations. The use of existing date formats can eliminate the need for massive system modifications. Methods of using existing date formats across century boundaries are explained. The use of a format termed the Lilian date format in honor of Luigi Lilio, the inventor of the Gregorian calendar, is introduced. The requirements for an effective date-processing algorithm are presented.

The Gregorian calendar serves us quite well in our day-to-day living. Due to discontinuities in various date divisions, however, it is not readily adaptable to computer programming. This fact becomes more apparent as we approach the new century. Few efficient, easy-to-use functions for manipulating dates have been produced. Also to be considered are the human requirements for ease of use, development, and maintenance. Other considerations include storage costs, efficiency, and adaptability across many different applications and environments.

Some early date-conversion programs were acts of expediency rather than planning, created to solve specific problems rather than for general programming use. Different functions were created at different times. Naming conventions, invocation formats, and implementation methods have often been inconsistent. Programs that provided a day-of-week function were rare. Also rare were programs for deriving a new date by adding or subtracting from another date in a different year. The functions that were available were normally not part of an integrated system for providing compatibility with most of the common date formats. Since documentation was not always created or maintained, individuals were often unaware of what was available. Today, dating format standards are being discussed internationally. The date-processing method presented in this paper is expected to be compatible with any foreseeable international standard date format as well as with the several formats discussed in this paper.

Typically, dates are displayed and stored in the *Julian* and *Gregorian* formats. Concepts and formats of both Julian and Gregorian date formats are discussed later in this paper. Simply stated, the Julian date is the number of the year together with the serial number of the day of that year. The Gregorian date format consists of month, day of month, and year. Many calendars show Julian dates printed in small numerals.

A format termed the *Lilian date format* is presented here as the basis for making date conversions of the

type mentioned earlier. The Lilian format, which may be stored in three bytes of memory, provides the storage capacity for dates well beyond the year 10 000. This format handles processing across century years and other aspects of date conversion not currently adaptable to computer programming. Practical date processing services must provide ease of use for the end user, developer, and maintainer. Such services must also eliminate date ambiguity, achieve excellent performance, and minimize storage.

The two positions traditionally used in both Julian and Gregorian date formats implicitly represent a year within a century. However, this system is inadequate for representing dates in more than one century. For example, it is ambiguous as to whether 03 represents the year 1903 or the year 2003. The Lilian date format avoids the ambiguity by using seven positions for the number of days from the beginning of the Gregorian calendar, October 15, 1582.

Origins of date complexity

Julian calendar. The Julian calendar was established in 46 B.C. by Julius Caesar.[1] It replaced a system of constant adjustments, more for political reasons than for correcting inaccuracies in timekeeping. After a bad start, with the first few leap years being calculated incorrectly, it served quite adequately for many centuries. The Julian calendar is not the same as the Julian date, which was previously defined. The Julian calendar was a time-measuring system, which we now discuss.

The scheme of the Julian calendar was quite simple. The calendar year consisted of 365-day years, with years evenly divisible by four having 366 days. The resulting average year of 365.25 days was slightly longer than the *tropical year*, which is based on the time marked by the passage of equinoxes. That slight difference resulted in a gradual drift of the calendar year. The resulting difficulty in properly setting the date of celebration of Easter caused great concern to the Roman Church. The date for Easter is related to the date for Passover, which is related to the vernal equinox. By the sixteenth century, the discrepancy approximated ten days, and the desire for calendar reform intensified. An artifact of this discrepancy manifests itself today in the differences in dates for Christmas and Easter between the Western Church and the Eastern Church. The latter continues to use the Julian calendar dates as they were at the time of the Gregorian calendar reform. That is, the Eastern Church recognizes the Gregorian calendar, but for certain feast days does not void the ten-day error that existed at the time of the calendar reform.

Gregorian calendar. For the calendar reform, Pope Gregory XIII selected[2] the plan of Aloysius Lilius

England and her colonies adopted the Gregorian calendar on Thursday, September 14, 1752.

(1510–1576?),[1] who is also known as Luigi Lilio.[3] This reform, known as the Gregorian calendar, was implemented on Friday, October 15, 1582.[2]

Although use of the Gregorian calendar spread rapidly among Roman Catholic countries, many centuries passed before it was generally used by non-Catholic countries. England and her colonies, including what is now the United States, adopted the calendar on Thursday, September 14, 1752.[3] Many other countries—notably China, Greece, and Russia—did not adopt it until after the beginning of the 20th century. In fact, Russia adopted the Gregorian calendar on two separate occasions, first in 1918, about the same time as many other countries, and again on June 27, 1940.[4] Changes made in Russia in 1929 to avoid the religious associations of the Gregorian calendar were unsuccessful, and it was reimplemented in 1940.

The reform that synchronized the calendar year with the tropical year required the removal of ten days from the calendar. As a result, Friday, October 15, 1582, immediately followed Thursday, October 4, 1582. (An alternate plan would have removed the ten excess days gradually by canceling leap days for 40 years.) As before, every fourth year is a leap year, and, to maintain synchronization, centurial years that are evenly divisible by 400 are also leap years.[2] Thus, for example, 1900 was a common year; the year 2000 will be a leap year, and the year 2100 will start a series of common centurial years again. The result is an average calendar year of 365.2425 days, which closely approximates the tropical year. Despite

this close approximation, by the 44th century,[5] the Gregorian calendar will again differ from the tropical year by one day. Because the tropical year consists of an irrational number of days, no calendar year with an integral number of days can exactly match the tropical year. Not only is there not an integral

Discontinuities in the Gregorian calendar scheme cause most of the difficulties with date manipulation.

number of days in a tropical year, but also the length of the day is not constant. That is, the length of the tropical year is also changing gradually.

Algorithms for date processing

Centurian date format. Discontinuities in the Gregorian calendar scheme cause most, if not all, of the difficulties associated with date manipulation. Instances of discontinuities are the following:

- No calendar unit is evenly divisible by weeks.
- The number of days per month varies.
- The number of days per year varies.
- The number of days per century varies.

These facts must be accounted for in calculating the number of days between two dates, calculating a date based on the addition or subtraction of days from another date, and calculating the day of the week for a date. A *centurian date format* (DDDDD, representing the day within a century) works well by giving continuous values within a century. Also, other date formats and day of week are readily calculable from this value. The centurian format is limited to a century and results in discrepancies when a century boundary is crossed. Consideration must be given to century years when making a leap year calculation. The DDDDD format will not span more than 273 years (or 179 years in a two-byte binary format). Also, a standard point of origin for the starting day must be defined.

Lilian date format. The Lilian date format consists of seven positions for the number of days from the beginning of the Gregorian calendar. Day one is Friday, October 15, 1582, and the value is incremented by one for each subsequent day. Based on this format are services supporting 44 experimental functions (or more if the three DMY, MDY, and YMD experimental versions of the Gregorian format are counted) synthesized from eight mnemonics.

Function-naming convention. Functions are named to promote easy recall and to suggest the activity to be performed. In the algorithm and experimental PL/I program discussed in this paper, each function name is synthesized from two 3-letter mnemonic parts. These mnemonic parts are defined as follows:

- CLL: compact Lilian date
- DAY: day of the week
- GRG: Gregorian date
- JUL: Julian date
- LIL: Lilian date
- SGR: short Gregorian date
- SJL: short Julian date
- VAL: validate a date

The first part is a description of the *target*, and the second part is a description of the *source*. VAL (validation) and DAY (day of week) can appear only in the target position of the function.

Arguments presented to any of the conversion functions are always presented in the same order: new date (target); old date (source); range date (control), when required; and Gregorian format [specifier(s)], when required. Table 1 illustrates the format for conversion arguments. In all instances, a return code is set.

The format descriptions in Table 1 refer to the type of data—D for day, M for month, and Y for year, as well as the number of positions (e.g., YY for two year positions)—that the data occupy. The origin of the term "Julian date" for the YYDDD format is given in Reference 6. Nevertheless, dates represented in the Julian format conform to the Gregorian calendar and not to the Julian calendar. The Julian date for Thursday, November 14, 1985, is 85318 (short form) and 1985318 (extended form).

Algorithms. The process of conversion between a Lilian format date and a Julian format date is now described. The algorithms are simpler to calculate by choosing a virtual base date rather than the real one. After the calculations are made, any discrepancies are resolved. In the following algorithms, the num-

Table 1 Examples of conversion function arguments

Argument	Description
target=JULSJL (source, 1925)	Conversion to a Julian (YYYYDDD) date from a short Julian (YYDDD) date within the range 1925–2024
target=JULGRG (source, DMY)	Conversion to a Julian (YYYYDDD) date from a Gregorian (DDMMYYYY) date
target=LILGRG (source, YMD)	Conversion of a Lilian (packed format) date from a Gregorian (YYYYMMDD) date
target=CLLJUL (source)	Conversion of a compact Lilian (binary format) date from a Julian date
target=SGRGRG (source, YMD, MDY)	Conversion of a short Gregorian (YYMMDD) date from a Gregorian (MMDDYYYY) date
CALL VALJUL (source)	Validation of a Julian date
target=DAYSJL (source 1925)	Day of week for a short Julian date

bers in parentheses are results of calculations made on the previously given date, November 14, 1985.

Extended Julian to Lilian. Given an extended Julian date (YYYYDDD), compute the corresponding Lilian date. First, compute the number of days from virtual January 1, 1501, to the start of the year being converted (1985318 Julian).

- Subtract 1501 from the Julian year (484).
- Multiply the difference by 365.25 (i.e., the average number of days per year) (176781.00).
- Truncate the result. The leap day is kept only for full four years (176781).

Then compute the number of Julian calendar (not Julian format) days from October 15, 1582, to the date being converted.

- Subtract 29872, i.e., the number of days between virtual January 1, 1501, and October 15, 1582 (146909).
- Add the Julian days (+318 = 147227).

Next, make the Gregorian calendar adjustments to this value.

- Subtract 1501 from the Julian year (484).
- Divide the difference by 100. One leap year is usually skipped each century (4.84).
- Truncate the result to obtain the number of whole leap days. Partial leap days do not exist (4).
- Subtract the number of whole leap days from the number of Julian calendar days (147223).

Because one out of every four centuries keeps all of its leap years, use the virtual year 1201, which is the beginning of the four-century cycle that contains the year 1582, to compute the number of leap years up to the target date.

- Subtract 1201 from the Julian year. The first four-hundred-year cycle began with virtual year 1201 and ended in the real 1600 (784).
- Divide the difference by 400 because one century leap year per 400 years is kept (1.96).
- Truncate the result because partial leap days do not exist (1).
- Add these leap days back into the adjusted Julian calendar days (147224).

This final result is the number of Gregorian calendar days from the beginning of the Gregorian calendar (October 15, 1582) to the date being converted. This result is the sought-for Lilian date.

Lilian to extended Julian. The purpose of this example is to show the procedure for converting a Lilian date to an extended Julian date. First, create a virtual Lilian date, with a starting point of January 1, 1201. Convert this result into a Julian calendar (not Julian format) date. Then convert the Julian calendar date to a Julian format date. The procedure is as follows (147224 Lilian):

- Add 139444 to the Lilian date. This is the number of days from virtual January 1, 1201 (the start of a 400-year cycle) to October 15, 1582. The result is a pseudo-Lilian date (286668).
- Divide the pseudo-Lilian date by 36524.25, the average number of days per century (7.85).
- Truncate the result to obtain the number of century leap days (7).

- Add the number of century leap days to the pseudo-Lilian date (286675).
- Divide the number of century leap days by 4 (1.75).
- Truncate the result to obtain the number of four-century leap days (1).
- Subtract the number of four-century leap days from the pseudo-Lilian date, because they are already included (286674).

The result is the number of full Julian calendar days from January 1, 1201, to the date being converted. We now convert this to an extended Julian format date.

- Divide full Julian calendar days by 365.25. This usually gives one less than the year for the date being converted (784.87).
- If there is no remainder from the division, subtract one from the quotient; otherwise, truncate the quotient to get the pseudo-Julian prior year (784).
- Multiply the pseudo-Julian prior year by 365.25 to determine the number of annual Julian calendar days prior to the year for the date being converted (286356.00).
- Truncate the number of annual Julian calendar days to eliminate partial leap days (286356).
- Subtract the number of truncated annual Julian calendar days from the full number of Julian calendar days to obtain the number of current Julian calendar days in the year of the date being converted (318).
- Add 1201 to the pseudo-Julian prior year to obtain the real Julian year, i.e., 1200 for years prior to 1200 plus one for the current year (+784 = 1985).
- Multiply the real Julian year by 1000 to put it into its proper Julian format position (1985000).
- Add the current Julian calendar days to the result to complete the Julian format date from the Lilian format date (1985318).

Ease of conversion

Accommodating end users. End users usually enter two digits for the year in a date and understand the ambiguity that this represents. Therefore, even at the turn of the century, to avoid adverse user reaction, programs must continue to function with only two digits for year. The inference of the year 1997 from 97 and 2003 from 03 must continue. For the exceptional case where the correct meaning could be 1897 and 1903, entry of all four digits may be required.

The month-day-year and day-month-year formats are ambiguous. Therefore, it might be advisable to continue presenting the date in its conventional U.S. format with a parenthetical explanation of the format—such as (MM/DD/YY)—to avoid the ambiguity.

It is not necessary to change date formats in files, because it is possible to change the programs only.

However, it may be necessary to provide a conversion function that receives a definition of the implied century as a parameter. An excellent way to do this unambiguously is to specify a year as the desired starting point of a 100-year range. For example, if the starting year for the range is specified as 1925, dates with year digits of 25 through 99 would be between 1925 and 1999, and dates with year digits of 00 through 24 would lie between 2000 and 2024.

Accommodating systems support. The conversion of isolated files to new date formats presents a rather trivial problem. In most cases, however, it is not possible to isolate the process. All programs that access the modified data must be changed simultaneously. In some large systems, literally thousands of programs may be involved. In these large systems, it may be prudent to avoid the cost and risk of massive changes in a short period of time.

It is not necessary to change date formats in files, because it is possible to change the programs only, so that the implied century in a date is recognized. Of course, in the vast majority of cases, that is exactly what does take place. Dates familiarly and implicitly exist within the 100-year range beginning with 1900 or 1901. Thus it is necessary merely to modify the programs so that the 100-year range starts at a later date. A beginning date set eighty years prior to the current systems date may be a reasonable convention. This is well within the range now in use.

The significant feature of this approach is that everything need not be modified at once. The modifications may be made over a period of years during

normal program maintenance. Of course, as systems are maintained or replaced, it would be practical to implement full information date formats. Where systems contain dates that span a range of more than 100 years, the century must already have been carried. In the rare event that this is not true, immediate conversion is unavoidable. Fortunately, in most applications, we can deal with centuries as exceptions rather than as a common problem.

Computational considerations

Storage. The two main considerations pertaining to storage are (1) the cost of storing large quantities of data; and (2) the computational cost of converting records within computer files to larger date-field sizes. When millions of dates are stored, as they are in most business systems, every additional byte required to save a single date multiplies to millions of additional bytes of storage. The programming necessary to accommodate larger date-field sizes in records further complicates date conversion.

Putting bounds on data-storage costs at reasonable levels and reducing conversion complications are both achievable. The allocation of additional space to record four positions for year, rather than the traditional two, is not the only possibility. Many systems currently store dates internally in packed Julian format, requiring three bytes of storage. In packed format, a Lilian date or an extended Julian date (YYYYDDD) both require four bytes of storage. However, if a binary format were to be adopted, either form of date could be stored in the three bytes used at present.

A more restrictive situation exists for systems in which dates are stored in two bytes instead of three. These systems use a binary format to record centurian or other forms of dates. In the near term for these systems, it may be necessary to continue storing dates in only two bytes. This cannot be accommodated with a Julian format. However, it is possible to store a range of dates by taking advantage of the continuous characteristic of a Lilian date.

Using the Lilian date system, any date in a selected range of 179 years may be stored as follows. Assume that the selected range is January 1, 1901, through December 31, 2079. Find the Lilian value of December 31, 1900. Subtract this value from the Lilian value of the date to be stored. Convert the result to a 16-bit (two-byte) binary value and store the result. Reverse the process to restore the two-byte date to Lilian format. Continuing to store dates in two bytes should be considered only where the programming cost of increasing record sizes in an existing system is prohibitive. The decreasing cost of storage makes the use of the full Lilian or Julian format a practical possibility when an application is being modified.

In the experiments on which this paper is based, the three-byte Lilian format for data storage has been found to be preferable. Internally, in computer use, the continuous nature of the Lilian date accommodates the types of processing that normally take place within a program. In addition, for all three storage sizes, a consistent format (i.e., the all-Lilian format or the quasi-Lilian format) is maintained.

The continuous nature of the Lilian date accommodates the types of processing that normally take place.

Flexibility. In our experiments, the IBM System 360/370 assembly language was used for coding the date functions. However, the method is easily adaptable to other programming languages such as COBOL, PL/I, and RPG. The experimental functions were also made re-entrant for flexibility within on-line environments.

Validity. Ideally, the validation of a date is necessary only at the point of its manual entry into a system. Experience teaches us that it is not unusual for an interfacing system to provide invalid dates. Therefore, all dates passed to conversion functions must be validated. When an invalid date is encountered by the experimental system, a null value is returned to the invoking program and a return code is set to indicate the nature of the invalid condition.

Efficiency. Date conversions are used heavily within certain applications. Because of the impact on a single system or an installation, efficiency must be inherent in date-conversion programs. Storage requirements for external display and internal calculations by computer programs are expected to mo-

tivate a high volume of conversions. Widely used programs in high-volume environments must perform well and not consume excessive amounts of storage. A date-conversion program that is good in all other ways will not be widely used if it is an operational bottleneck.

For the program discussed in this paper, written in System 360/370 assembly language and implemented as a set of PL/I functions, the experimental results indicated good efficiency. The longest execution paths, including PL/I prologue, validation, conversion, and PL/I epilogue, are a little more than one hundred machine-language instructions. Although the functions appear to the invoking PL/I program to be separate programs, they are all actually provided within a single program that requires less than 4K bytes of storage.

Experimental results indicated good efficiency.

Concluding remarks

Programmers require date-processing functions that effectively handle applications for both the present and the future. For the present, it is sufficient to validate source dates, to convert from one traditional date format to another, and to perform addition or subtraction operations involving dates. For the future, additional functions are required that support processing restricted only by the limitations of the Gregorian calendar. These functions must be fully compatible with existing date-format and record-size restrictions. Massive conversion efforts should not be required to process and store dates outside the twentieth century. Also, end users should be able to continue to use existing two-digit date formats when interacting with computer systems. In all cases the programs that provide these services must be reliable, efficient in the use of both processor and storage, and flexible in application.

Programs that embody all these qualities have been written and tested experimentally. The application as written requires less than 4K bytes of storage and has an average execution path of fewer than 100 machine language instructions. Re-entrancy and the possibility of multilanguage implementation indicate excellent flexibility. This approach presents a practical method of processing dates that is compatible with any dating format standard.

Acknowledgments

No significant work is done in isolation. Over the years, I have experienced the inspiration, assistance, and patience of many individuals. My colleagues Tom Gauthier and Jim Willard first sparked my interest in date processing several years ago. Recently Alex Chang, Barb Henderson, Jack Henriksen, Fred Lange, Bob Lord, Libby Ross, Karen Seabury, and Billy Shih have patiently served as a sounding board, made helpful suggestions, and have been active supporters. Sandy Mink provided valuable editorial support. Many unnamed others have been involved in my experiment, including every member of my family. Their support is also greatly appreciated.

Cited references

1. G. Moyer, "Luigi Lilio and the Gregorian reform of the calendar," *Sky and Telescope* **64**, No. 11, 418–419 (November 1982).
2. J. J. Bond, *Handy Book of Rules and Tables for Verifying Dates Within the Christian Era*, Russell & Russell, a Division of Atheneum House, Inc., New York (1966).
3. *The New Encyclopedia Britannica*, Encyclopedia Britannica, Inc., Chicago (1980), p. 602.
4. F. Parise, Editor, *The Book of Calendars*, Facts on Life, Inc., New York (1982).
5. G. Moyer, "The Gregorian calendar," *Scientific American* **246**, No. 5, 144–152 (May 1982).
6. M. A. Covington, "A calendar for the ages," *PC Tech Journal* **3**, No. 12, 136–142 (December 1985). Joseph Justice Saliger (1540–1609) named the Julian date in honor of his uncle Julius.

General references

G. Moyer, "Astronomical scrapbook—Notes on the Gregorian calendar reform," *Sky and Telescope* **64**, No. 12, 530–533 (December 1982).

International Organization for Standardization, "Writing of calendar dates in all-numeric form," Reference No. ISO 2014-1976(E) (April 1976).

International Organization for Standardization, "Information processing interchange—Representation of ordinal dates," Reference No. ISO 2711-1973(E) (January 1973).

Bruce G. Ohms *IBM Information Systems Group, 301 Merritt 7, Norwalk, Connecticut 06856.* Mr. Ohms joined IBM in 1967 and is currently a senior programmer/analyst working in the area of applications systems development. Prior to his current assignment, he has held a variety of positions in programming, systems design, analysis, and development center implementations. He received an A.A.S. degree in data processing from Belleville Area College, Belleville, Illinois, and he attended Yale University.

Reprint Order No. G321-5274.

Japanese Published Application 05-027947,

February 5, 1993

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号

特開平5－27947

(43)公開日　平成５年(1993)２月５日

(51)Int.Cl.[5]	識別記号	庁内整理番号	Ｆ Ｉ	技術表示箇所
Ｇ０６Ｆ　7/24		8323－5B		
15/02	３３０　Ｐ	9194－5L		

審査請求　未請求　請求項の数１(全　４　頁)

(21)出願番号　特願平3－177982

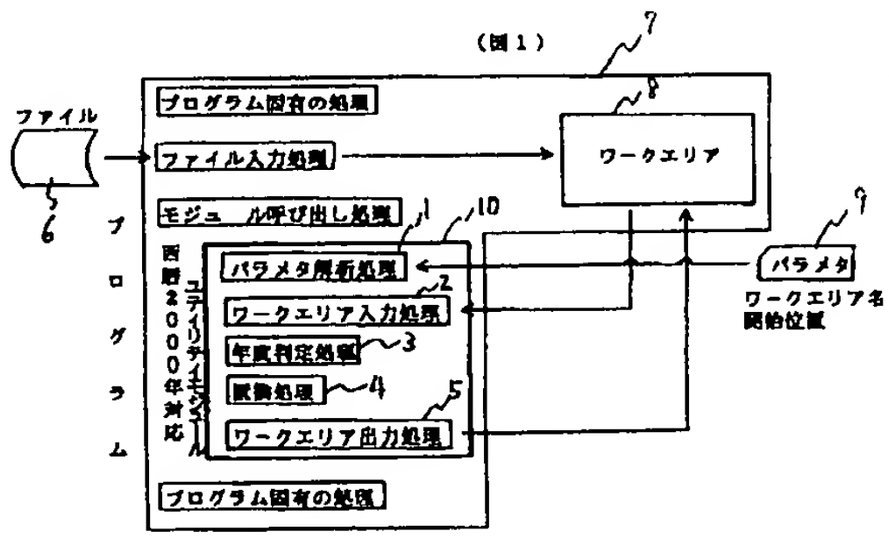
(22)出願日　平成３年(1991)７月18日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(72)発明者　狭間　正和
神奈川県川崎市幸区鹿島田890番地の12
株式会社日立製作所情報システム開発本部内
(74)代理人　弁理士　小川　勝男

(54)【発明の名称】　年度順保証方法

(57)【要約】
【目的】データファイル中の西暦の下２ケタで年度を管理している場合でも、現状のデータファイルフォーマットで、西暦２０００年以降も年度順を保証する。
【構成】年度の大小判定，ソート・マージ処理の前に、西暦２０００年対応ユティリティモジュール１０を起動する。その後、外部パラメタ９により予め西暦下２ケタが格納されているレコード中の位置を指定し、予め指定した範囲の西暦を示すコードを、年度の順序が維持される様に他のコードを置き換える。

(図１)



【特許請求の範囲】
【請求項１】記憶手段と処理部を有するコンピュータシステムにおいて、前記記憶手段に１９００年代と２０００年代の西暦の下２桁が格納されている場合、処理部は、西暦の下２桁のうち十の位のコードを、年度の順序を維持するコードに置き換えることを特徴とする年度順保証方法。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】本発明は、西暦の下２桁が格納されているデータファイルに対する、プログラムによる大小判定及びソート・マージ処理での昇降順を保証し、正常な結果を得る西暦２０００年に対応した年度順保証方法に関するものである。
【０００２】
【従来の技術】従来のコンピュータシステムでは、データファイル中に、西暦の下２桁が格納されている。なお、この種の技術に関するものとしては、例えば、特開昭５８-１２２９号公報や、特開平３-２２１１７号公報がある。
【０００３】
【発明が解決しようとする課題】上記従来技術は、西暦２０００年以降の場合を考慮しておらず、データファイル中の西暦の下２桁で年度を管理している。そのため、西暦２０００年以降では、通常の年度による大小判定処理及びソート・マージ処理での昇降順序処理を行った場合、その大小関係は、数１で表わされる。
【０００４】
【数１】

$$19\underline{99} > 19\underline{98} > 20\underline{01} > 20\underline{00} \quad \cdots (数1)$$

【０００５】つまり、２０００年は、１９９９年よりも大きいと判断されなければならないにも関わらず、判断にあたっては、西暦の下２桁しか用いないので、００は９９より小さいことから、２０００年は１９９９年よりも小さいと判断されてしまう。これを解決する方法として、西暦を４桁にすることが考えられるが、この場合、データファイルのレコード長，ブロック長を変更することが必要となり、また、プログラム修正も発生してしまうという問題点がある。
【０００６】本発明の目的は、コードの体系を生かし、現状のデータファイルフォーマットで西暦２０００年以降も対応可能な西暦２０００年に対応した年度順保証方法を提供することにある。
【０００７】
【課題を解決するための手段】上記目的を達成するために、記憶手段と処理部を有するコンピュータシステムにおいて、前記記憶手段に１９００年代と２０００年代の西暦の下２桁が格納されている場合、処理部は、西暦の下２桁のうち十の位のコードを、年度の順序を維持するコードに置き換えることにしたものである。
【０００８】
【作用】１９００年代と２０００年代の西暦を示すデータが混在する場合、年度の順序が維持される様に、西暦を示すデータのコードを他のコードと置き換える。これにより、大小判定、昇降順序処理が保証される。
【０００９】
【実施例】図１は、本発明の一実施例のプログラムの構成図である。
【００１０】６は、下２桁のみの西暦を含むデータが格納されたファイルである。７は、プログラムである。８は、ワークエリアである。９は、パラメタである。１０は、西暦２０００年対応ユティリティモジュールである。
【００１１】西暦２０００年対応ユティリティモジュール１０（以下モジュール１０とする）は、プログラム７やユティリティで指定されると動作し、年度を取扱う処理の前処理として位置付けられる。なお、モジュール１０には、予め、コード変換を行なう下２桁の範囲を指定しておく。置き換えるのは、１９００年代で下２桁が一番小さい数よりも下２桁が小さい２０００年代の数である。例えば、ファイル６中のデータの西暦が１９７３年から始まる場合は、下２桁が００（２０００年）から７２（２０７２年）までを変換する。本実施例は、ファイル６には１９６０年からのデータが有る場合の例でなので、下２桁が００～５９の場合にコード変換をするように範囲を指定する。なお、本実施例では、コードとしてＥＢＣＤＩＣコードを用いている。
【００１２】次に、処理手順を説明する。まず、プログラム７は、年度判定を行う前段階でモジュール１０の呼び出し処理を行う。その後、モジュール１０では以下の処理を行う。
【００１３】パラメタ解析処理１：外部より与えられたパラメタ９を入力して内容を解析し、ワークエリア名及びレコード中の西暦下２桁の開始位置を認識する。
【００１４】ワークエリア入力処理２：パラメタ解析処理１で得たワークエリア名からデータを入力する。
【００１５】年度判定処理３：パラメタ解析処理１で得た、レコード中の西暦下２桁の開始位置からの２バイトを判定し、一定の範囲内の時、本実施例では’００’から’５９’の範囲の場合に置換処理４を行う。それ以外は、次のデータを入力する。
【００１６】置換処理４：レコード中の西暦下２桁のうち十の位を表１の様に置換する。
【００１７】

【表1】

（表1）

	置換前	置換後
十の位が0の場合	X' F0'	X' FA'
十の位が1の場合	X' F1'	X' FB'
十の位が2の場合	X' F2'	X' FC'
十の位が3の場合	X' F3'	X' FD'
十の位が4の場合	X' F4'	X' FE'
十の位が5の場合	X' F5'	X' FF'

【0018】ワークエリア出力処理5：置換処理4を行ったデータをワークエリア8に出力する。

【0019】表1に示される様に、文字コードを置き換えることにより、1990年よりも2000年が、2000年代より2010年が大きいと判断される。

【0020】なお、本実施例では、EBCDIKコードで、2000年以上の場合、表1に示すコードの置き換え処理を行ったが、2000年未満の場合に、表2の様に、空いているコードと置き換えてもよい。このやり方では、例えば、ファイル6に1999年からのデータが有る場合ならば、2098年まで対応できる。

【0021】

【表2】

（表2）

	置換前	置換後
十の位が0の場合	X' F0'	X' B0'
十の位が1の場合	X' F1'	X' B1'
十の位が2の場合	X' F2'	X' B2'
十の位が3の場合	X' F3'	X' B3'
十の位が4の場合	X' F4'	X' B4'
十の位が5の場合	X' F5'	X' B5'
十の位が6の場合	X' F6'	X' B6'
十の位が7の場合	X' F7'	X' B7'
十の位が8の場合	X' F8'	X' B8'
十の位が9の場合	X' F9'	X' B9'

【0022】また、2000年未満の場合に、X'F0'→X'C0'等、使用しているコードと置き換えてもよい。

【0023】また、2000年代の場合に、X'F0'→X'C0'と、1900年代の場合に、X'F0'→X'B0'と置き換えてもよい。つまり、2000年代と1900年代の両方を他のコードに置き換えてもよい。

【0024】また、使用する文字コードはJISコードやASCIIコード等でも構わない。　つまり、年度の大小関係等を判断するときに、判断が正常に行える様にコードの置き換えを行えばよい。

【0025】

【発明の効果】年度の大小関係判定が正常に行える様にコードの置き換えを行うことにより、データファイルのレコード長，ブロック長を変更せず、またプログラム修正もせずに年度の順序が保証されるという効果がある。

【0026】

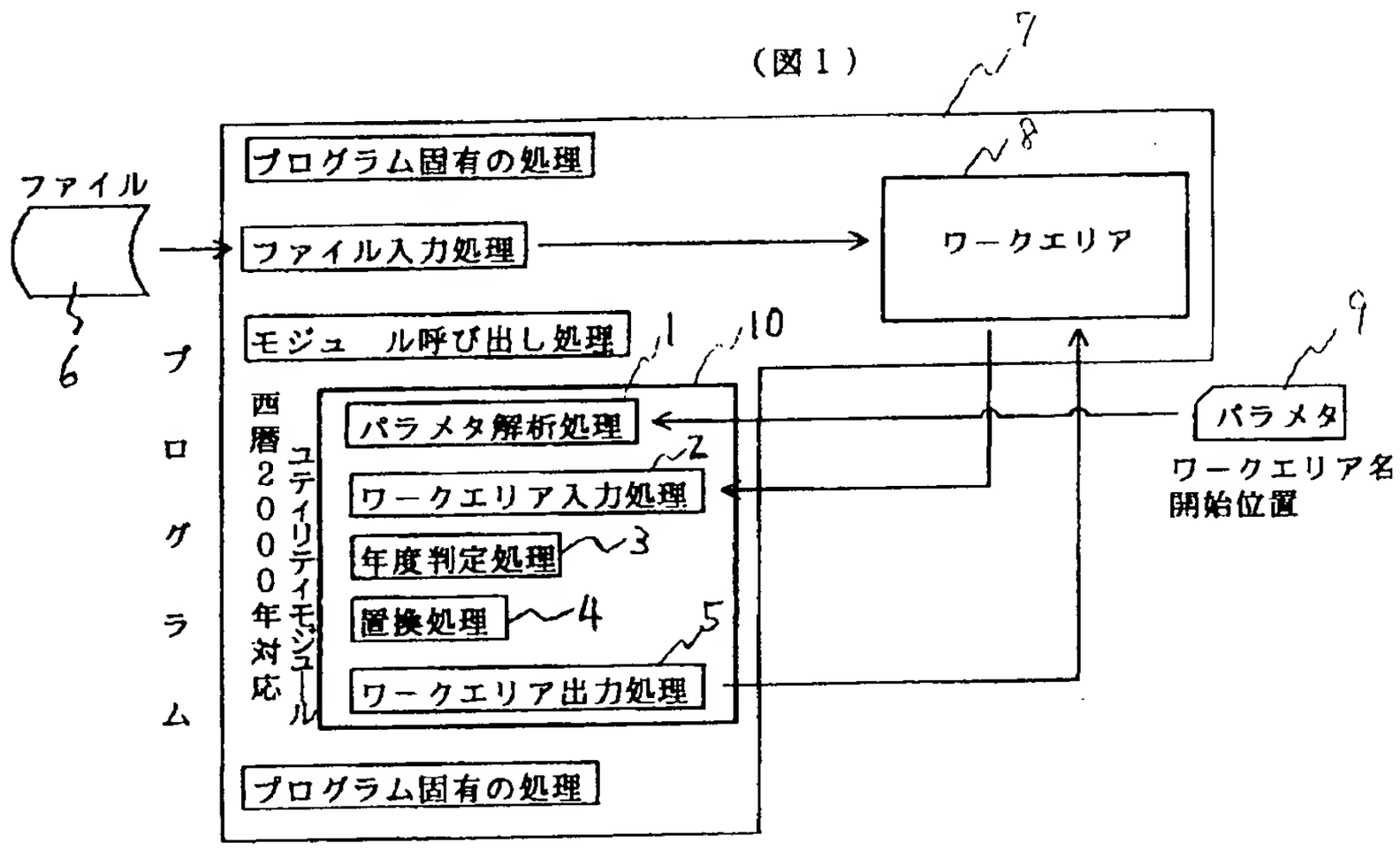
【図面の簡単な説明】

【図1】プログラムの構成を示す図である。

【符号の説明】
1…パラメタ解析処理、2…ワークエリア入力処理、3…年度判定処理、4…置換処理、5…ワークエリア出力処理、6…ファイル、7…プログラム、8…ワークエリア、9…パラメタ、10…西暦2000年対応ユティリティモジュール。

【図1】

（図1）



Japanese Published Application 05-027947,

February 5, 1993

Translation

Japanese Kokai Patent Application No. Hei 5[1993]-27947

Translated from Japanese by the Ralph McElroy Translation Company
910 West Avenue, Austin, Texas 78701 USA

Code: 1058-74331
Ref: 1968-7378

JAPANESE PATENT OFFICE
PATENT JOURNAL (A)
KOKAI PATENT APPLICATION NO. HEI 5[1993]-27947

Int. Cl.[5]:	G 06 F 7/24 15/02
Sequence Nos. for Office Use:	8323-5B 9194-5L
Filing No.:	Hei 3[1991]-177982
Filing Date:	July 18, 1991
Publication Date:	February 5, 1993
No. of Claims:	1 (Total of 4 pages)
Examination Request:	Not filed

METHOD OF GUARANTEEING YEAR ORDER

Inventor:	Masakazu Hazama Hitachi, Ltd., Information Systems Development Headquarters 890-12 Kashimada, Saiwai-ku, Kawasaki-shi, Kanagawa-ken
Applicant:	000005108 Hitachi, Ltd. 6 Surugadai, 4-chome, Kanda, Chiyoda-ku, Tokyo-to
Agent:	Katsuo Ogawa, patent attorney

[There are no amendments to this patent.]

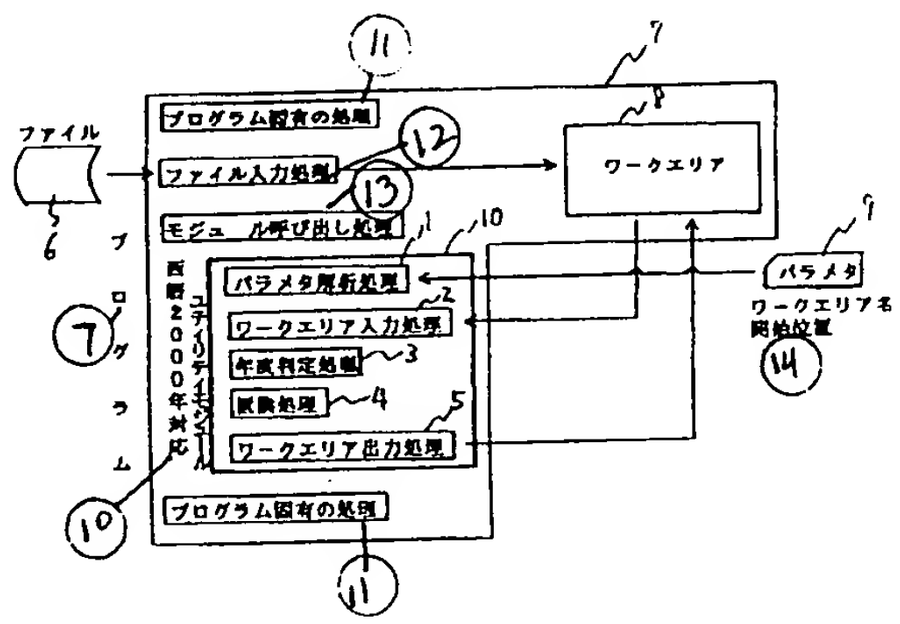
Abstract

Purpose

To guarantee the year order, even for years after 2000 AD, with the current file format, even when the year is managed by the last 2 digits of the date in digital files.

Constitution

Before the magnitude of the year is evaluated and before sort/merge processing, 2000 AD correspondence utility module (10) is activated. After that, the position in the record where the last 2 digits AD have been previously stored are specified by external parameter (9), and a code that represents a pre-defined range of dates is replaced by another code so that the year order will be maintained.



Key:

1	Parameter analysis processing
2	Work area input processing
3	Year evaluation processing
4	Replacement processing
5	Work area output processing
6	File
7	Program
8	Work area
9	Parameter
10	Year 2000 date correspondence utility module
11	Processing unique to program
12	File input processing
13	Module call-up processing
14	Work area name start position

Claim

1. Method of guaranteeing year order characterized in that, in a computer system that has a memory means and a processing section, when the last 2 digits for years in the 1900's and 2000's AD are stored in the aforementioned memory means, the processing section replaces the

code for the 10's place in the last 2 digits of the year AD with a code that maintains the year order.

Detailed explanation of the invention

[0001]

Industrial application field

This invention pertains to a method of guaranteeing year order for the year 2000 AD that guarantees ascending/descending order with evaluation of magnitude and sort/merge processing using programs for digital files in which the last 2 digits of the year AD are stored, and that can obtain correct results.

[0002]

Prior art

With conventional computer systems, the last 2 digits of years AD are stored. Note that, for example, Japanese Kokai Patent Application Nos. Sho 58[1983]-1229 and Hei 03[1991]-22117 involve this type of technology.

[0003]

Problems to be solved by the invention

The aforementioned prior art does not take into consideration years after 2000 AD, and manages the year with the last 2 digits of the year AD in a file. For this reason, after 2000 AD, when ascending/descending order is handled by processing that evaluates magnitude and by sort/merge processing using normally numbered years, their relative magnitudes are represented by formula 1.

1999 > 1998 > 2001 > 2000

[0005]

That is, regardless of the fact that the year 2000 must be evaluated to be larger than the year 1999, for evaluation, only the last 2 digits of the date are used, so since 00 is smaller than 99, year 2000 is evaluated to be smaller than year 1999. Using 4 digits for the date has been considered as a method of resolving this, but in this case, it is necessary to change the data file record length and block length, and program modifications also arise.

[0006]

The purpose of this invention is to provide a method of guaranteeing year order for handling 2000 AD that makes use of the code system and that can also handle years after 2000 AD.

[0007]

Means to solve the problems

To accomplish the aforementioned purpose, in a computer system that has a memory means and a processing section, when the last 2 digits for years in the 1900's and in the 2000's AD are stored in the aforementioned memory means, the processing section will replace the code of the 10's place in the last 2 digits of the date with a code that maintains the year order.

[0008]

Function

When there are data present that indicate years in the 1900's and 2000's AD, the data code that represents the date is replaced by another code so that the year order will be maintained. In this way, magnitude evaluation and ascending/descending order processing are guaranteed.

[0009]

Application example

Figure 1 is a block diagram of a program in one application example of this invention.

[0010]

(6) is a file where data that include only the last 2 digits of the year AD are stored. (7) is a program. (8) is a clear area. (9) is a parameter. (10) is a 2000 AD correspondence utility module.

[0011]

2000 AD correspondence utility module (10) (hereafter called module (10)) is activated when specified by program (7) or a utility and is positioned as pre-processing for processing that handles the year. Note that, in module (10), a range of the last 2 digits for which code transformation will be performed are specified in advance. Replacement involves numbers for years in the 2000's where the last 2 digits are smaller than the smallest number in the last 2 digits in years in the 1900's. For example, when data in file (6) for years AD begin with the year 1973, the last 2 digits are replaced using 00 (year 2000) for 72 (year 2072). The present application example is an example where there are data from year 1960 in file (6), such a range is specified

so that the last 2 digits will be transformed to codes 00-59. Note that, in the present application example, EBCDIC code is used as the code.

[0012]

Next, the processing sequence will be explained. First, program (7) calls module (10) at a preliminary stage that evaluates the year. After that, the following processing is performed by module (10).

[0013]

Parameter analysis processing (1): parameter (9), provided from outside, is input and the contents are analyzed, and the starting positions of the work area name and the last 2 digits of the year AD in the record are confirmed.

[0014]

Work area input processing (2): data from the work area name obtained by parameter analysis processing (1) are input.

[0015]

Year evaluation processing (3): 2 bytes from the starting position of the last 2 digits in the year AD in the record obtained by parameter analysis processing (1) are evaluated, and if within a fixed range, in the present application example, in the range from '00' to '59,' replacement processing (4) is performed. In addition to this, the next data are input.

[0016]

Replacement processing (4): the 10's place in the last 2 digits in the year AD in the record is replaced as in Table 1.

[0017]

Table 1

	置換前(1)	置換後(2)
(3) 十の位が０の場合	X' F 0'	X' F A'
十の位が１の場合	X' F 1'	X' F B'
十の位が２の場合	X' F 2'	X' F C'
十の位が３の場合	X' F 3'	X' F D'
十の位が４の場合	X' F 4'	X' F E'
十の位が５の場合	X' F 5'	X' F F'

Key: 1 Before replacement
2 After replacement
3 When 10's place is __

[0018]

Work area output processing (5): data that have undergone replacement processing (4) are output to work area (8).

[0019]

By replacing character codes as shown in Table 1, it is evaluated that year 2000 is greater than year 1990, and that year 2010 is greater than year 2000.

[0020]

Note that, in the present application example, the replacement processing with codes shown in Table 1 was performed for [years] greater than year 2000 with EBCDIK code, but in the case of [years] less than year 2000, they could also be replaced by empty code as in Table 2. With this method, for example, if there are data from year 1999 in file (6), up to year 2098 can be handled.

[0021]

Table 2

	置換前(1)	置換後(2)
(3) 十の位が０の場合	X' F 0'	X' B 0'
十の位が１の場合	X' F 1'	X' B 1'
十の位が２の場合	X' F 2'	X' B 2'
十の位が３の場合	X' F 3'	X' B 3'
十の位が４の場合	X' F 4'	X' B 4'
十の位が５の場合	X' F 5'	X' B 5'
十の位が６の場合	X' F 6'	X' B 6'
十の位が７の場合	X' F 7'	X' B 7'
十の位が８の場合	X' F 8'	X' B 8'
十の位が９の場合	X' F 9'	X' B 9'

Key: 1 Before replacement
2 After replacement
3 When 10's place is __

[0022]

Also, for [years] less than year 2000, they could also be replaced with code that uses X'F0' → X'B0', etc.

[0023]

Also, in the case of years in the 2000's, they could be replaced by X'F0' → X'C0', and for years in the 1900's, replaced by X'F0' → X'B0'. That is, both years in the 2000's and in the 1900's could be replaced by other codes.

[0024]

Also, it makes no difference if the character code used is JIS code or ASCII code, etc. That is, when the relative magnitudes of years are evaluated, code replacement need only be performed so that evaluation is correctly accomplished.

[0025]

Effect of the invention

By replacing code so that the relative magnitudes of years are correctly evaluated, the effect is that year order will be guaranteed without changing data file record length or block length, and further, without modifying programs.

Brief description of the figures

Figure 1 is a figure that shows program configuration.

Explanation of symbols

(1) ... parameter analysis processing, (2) ... work area input processing, (3) ... year evaluation processing, (4) ... replacement processing, (5) ... work area output processing, (6) ... file, (7) ... program, (8) ... work area, (9) ... parameter, (10) ... 2000 AD correspondence utility module.

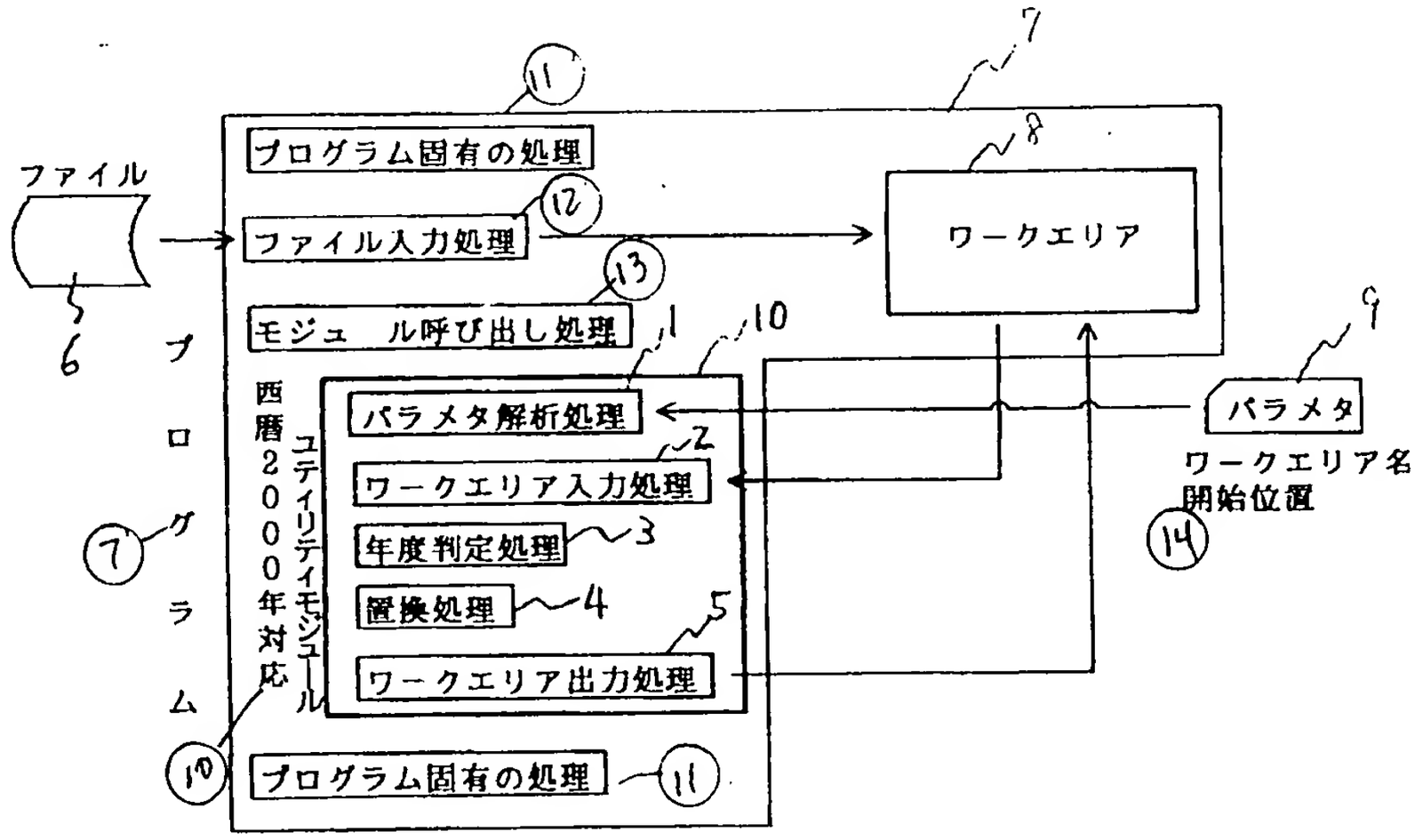


Figure 1

Key: 1 Parameter analysis processing
2 Work area input processing
3 Year evaluation processing

4	Replacement processing
5	Work area output processing
6	File
7	Program
8	Work area
9	Parameter
10	Year 2000 date correspondence utility module
11	Processing unique to program
12	File input processing
13	Module call-up processing
14	Work area name start position

(19) JAPANESE PATENT OFFICE

(11) Publication Number: --
JP 06103133 A

(43) Date of publication:
19940415

(51) int. Cl : G06F012-00
*

* (71) Applicant:
PFU LTD

* (72) Inventor:
SAKA NOBUO

(21) Application Information:
19920918 JP 04-249902

YEAR AND DATE KEY MANAGING METHOD FOR DATA FILE

* (57) Abstract:

PURPOSE: To provide the managing method for a year and date key being useful for processing a data file in which the year and date keys of both the years are mixed especially in the case the Christian Era year is transferred from the 1900 year level to the 200 year level, with regard to the managing method for the year and date key in a data processing system for processing the data file having the year and date key of an abbreviated form in which the upper two digits of the Chastian Era year are omitted.

CONSTITUTION: When a key file 3 of a year and date key is prepared with regard to a data file 1, or data are sorted by using the year and date key, year data in the year and date key is compared with a prescribed threshold, and based on its result, a value of two digits 19 or 20 is added to the higher coder of the year data, and the year data are restored to the Christian Era year of four digits. By the year and date key containing this restored year data, the key file 3 is prepared or the data is sorted.

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平6－103133

(43)公開日　平成６年(1994)４月15日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　12/00	５２０　Ａ	8526－5B		

審査請求　未請求　請求項の数１(全　４　頁)

(21)出願番号　特願平4－249902

(22)出願日　平成４年(1992)９月18日

(71)出願人　000136136
株式会社ピーエフユー
石川県河北郡宇ノ気町字宇野気ヌ98番地の２

(72)発明者　坂　信夫
石川県河北郡宇ノ気町字宇野気ヌ98番地の２　株式会社ピーエフユー内

(74)代理人　弁理士　長谷川　文廣　(外２名)

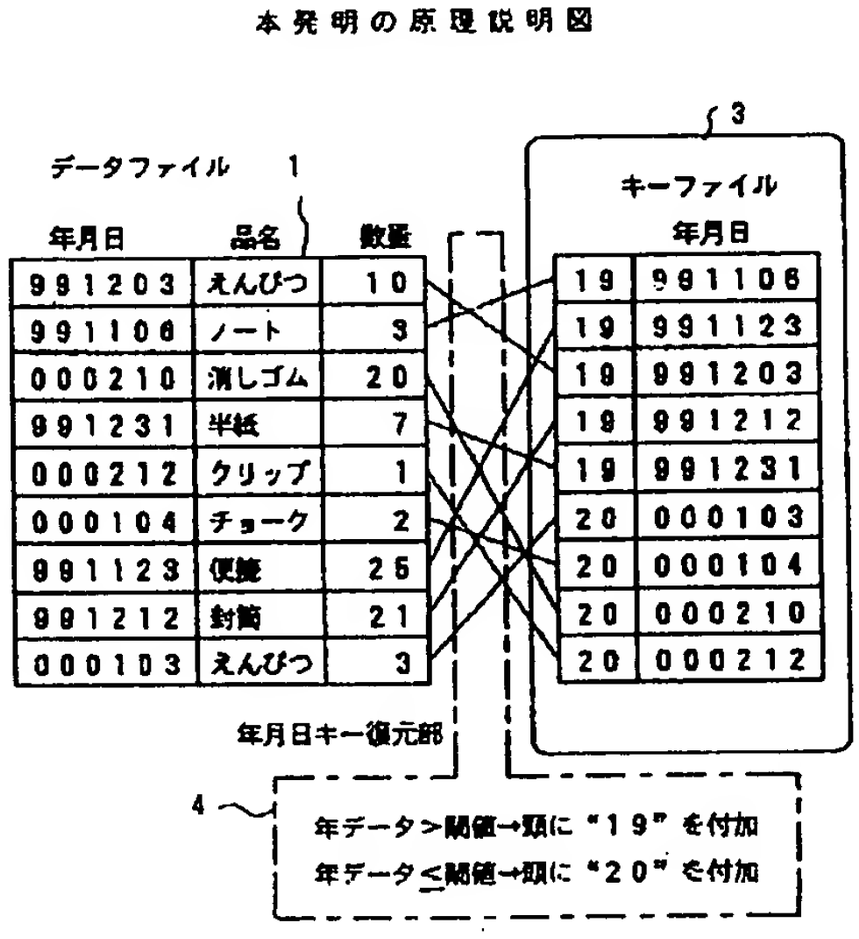
(54)【発明の名称】　データファイルの年月日キー管理方法

(57)【要約】

【目的】　西暦年の上位２桁を省略した短縮形の年月日キーをもつデータファイルを処理するデータ処理システムにおける年月日キーの管理方法に関するものであり，特に西暦年が１９００年台から２０００年台に移行した場合に，双方の年台の年月日キーが混在しているデータファイルを処理するために有用な年月日キーの管理方法を提供することを目的としている。

【構成】　データファイルについて年月日キーのキーファイルを作成，あるいは年月日キーを用いてデータのソートを行う際，年月日キー中の年データを所定の閾値と比較し，その結果に基づいて年データの上位に２桁の値１９あるいは２０を付加して年データを４桁の西暦年に復元し，この復元された年データを含む年月日キーにより，キーファイルの作成あるいはデータのソートを行わせる構成をもつ。

本発明の原理説明図



1

【特許請求の範囲】

【請求項1】　西暦年の上位2桁を省略し下位2桁のみを年月日キーの年データに用いるデータファイルを備えたデータ処理システムにおいて，上記データファイルについて年月日キーのキーファイルを作成，あるいは年月日キーを用いてデータのソートを行う際，年月日キー中の年データを所定の閾値と比較し，その結果に基づいて年データの上位に2桁の値19あるいは20を付加して年データを4桁の西暦年に復元し，この復元された年データを含む年月日キーにより，キーファイルの作成あるいはデータのソートを行わせることを特徴とするデータファイルの年月日キー管理方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は，西暦年の上位2桁を省略した短縮形の年月日キーをもつデータファイルを処理するデータ処理システムにおける年月日キーの管理方法に関するものであり，特に西暦年が1900年台から2000年台に移行した場合に，双方の年台の年月日キーが混在しているデータファイルを処理するために有用な年月日キーの管理方法を提供するものである。

【0002】

【従来の技術】通常，ファイル上で年月日データを管理する場合に，年については西暦年の上位2桁を省略した短縮形が用いられている。これは現在までのところ西暦年の上位2桁の値は“19”であって不変であるため，データが冗長になるからである。しかし西暦2000年を越えてデータを管理する状況になると，短縮形の年データは“00”となり，年月日をキーとする処理を行った場合，データの配列順序に不都合が生じる。

【0003】図3に従来例を示す。図において，1は，商品管理のデータファイルであり，各レコードは，購入日を示す年月日データと，品名データ，数量データからなる。年月日データは，たとえば“991203”は1999年12月3日を表わし，“000210”は2000年2月10日を表わす。このデータファイル1内のレコードの配列は，キーとなる年月日データの値に関してランダムになっている。2は，データファイル1の各レコードの年月日データについて作成したキーファイルであり，各年月日データが，値の大きさにしたがって配列されている。

【0004】この従来例のキーファイル2では，年月日データの最上位2桁の値が“00”のキーが“99”のキーに先行して配列されるため，境界で時系列上の逆転が生じ，具合が悪いものとなっている。一般に年月日データをソートする場合には，このような問題が生じる。

【0005】

【発明が解決しようとする課題】本発明は，西暦年の下位2桁を用いてキーとなる年月日データを構成しているデータファイルにおいて，西暦年が2000年を越えても，データファイルを変更する必要なしに時系列の逆転のないキーファイルの作成を可能にすることを目的としている。

2

【0006】

【課題を解決するための手段】本発明の年月日キー管理方法は，西暦年の下位2桁を年データとしているデータファイルにおいて年月日をキーとしてキーファイルを作成あるいはソートを行う際に，年データの2桁の値が西暦1900年台と西暦2000年台のいずれに属するものであるかを判定して，その結果に基づいて2桁の年データの頭に2桁の値“19”または“20”を付加して拡張し，4桁の西暦年に復元して，この復元した西暦年を含む年月日データによりキーファイルの作成あるいはソートを行うようにするものである。

【0007】この本発明による年月日キーの管理方法は，データファイルの2桁の年データについて，西暦1900年台のものの値の範囲と西暦2000年台のものの値の範囲とが通常は明確に分れており，適当な閾値を用いることによりそれぞれの値の範囲を容易に切り分けできることを原理としている。

【0008】図1は本発明の原理説明図である。図において，1は，データファイルであり，その各レコードは西暦年の下位2桁を年データとして含む年月日キーをもつ。

【0009】3は，キーファイルであり，年月日キー復元部4により復元された4桁の西暦年データを含む年月日キーを用いて作成されたものである。4は，年月日キー復元部であり，年データの2桁の値を所定の閾値と比較して，年データが閾値よりも大きい場合には頭に“19”を付加し，年データが閾値よりも小さい場合には頭に“20”を付加して，4桁の西暦年データを含む年月日キーを復元する。

【0010】

【作用】図1を用いて本発明の作用を説明する。図1のデータファイル1に例示されている各レコードの年月日データ（キー）は，頭の2桁が“99”か“00”である。このことは，対応する西暦年が1999年か2000年であることを意味する。これは，データファイル1には2099年のデータや1900年のデータが存在しないことが裏付けとなっている。この場合，閾値として“99”と“00”の中間の任意の値を使用し，年月日キー中の年データと比較することによって，年月日キーが1999年に属するか2000年に属するかを簡単に識別することができる。たとえば閾値を“50”に設定し，年データが“50”よりも大きい“99”であれば頭に“19”を付加し，また年データが“50”よりも小さい“00”であれば頭に“20”を付加すればよい。

【0011】一般に，1900年台の年データの最小値が“n_0 n_1”であり，2000年台の年データの最大

値が“n_2 n_3”であれば，閾値には“n_0 n_1”と“n_2n_3”の間の適当な値が使用される。

【００１２】このようにして復元された４桁の西暦年データをもつ年月日キーの値は時系列を正しく反映できるものとなるので，そのソートにより作成されるキーファイル３の各年月日キーは，図示のように時系列的に整列されたものとなる。

【００１３】

【実施例】図２に，本発明実施例によるデータ処理システムの構成を示す。図２において，データファイル１，キーファイル３，年月日キー復元部４，はそれぞれ図１で説明したものと同じである。また５はキーファイル作成部であり，年月日キー復元部４により復元された年月日キーに基づいてキーファイル３を作成する。６，７，８，９はシステムのハードウェア構成を示し，６は処理装置，７はディスプレイ，８はキー入力装置，９はプリンタである。

【００１４】年月日キー復元部４およびキーファイル作成部５の機能は，プログラムにより実現される。図示の例では，キーファイル作成部５は年月日キー復元部４に縦続的に接続され，年月日キー復元部４がデータファイル１から読み出し復元した年月日キーを受け取ってキーファイル３を作成する構成となっているが，キーファイル作成部５がデータファイル１から年月日キーを読み出し，年月日キー復元部４を呼び出して復元処理を依頼するように構成してもよく，またキーファイル作成部５が年月日キー復元部４の機能を内部に取り込んだ構成とすることも可能である。さらに年月日キー復元部の機能をもつプログラムをサポートプログラムとしてシステム側が備えていても，あるいはアプリケーションプログラム側が用意するようにしてもよい。

【００１５】図２の構成では，年月日キー復元部４は次のように動作する。まずデータファイル１から１つのレコードを取り出し，その年月日キー中の２桁の年データ（ｎｎ）を読む。次に年データ（ｎｎ）を予め設定されている閾値（α）と比較し，ｎｎ＞αであれば年データ（ｎｎ）の頭に“１９”を付加し，その他の場合には“２０”を付加する。このようにして４桁の年データに復元し，残りの月日データと合成してキーファイル作成部５に渡す。これらの処理をデータファイル１の順次のレコードについて実行し，最終レコードまで処理したときに終了する。

【００１６】キーファイル作成部５は，年月日キー復元部４から受け取った全ての年月日キーを図示されていないメモリの作業領域に書き込んで，重複するキーを除き，ソート処理を行って，たとえば昇順に配列したキーファイル３を作成する。

【００１７】本発明はキーファイルの作成を例に説明されたが，これに限定されるものではなく，年月日キーのソートを伴う任意のファイル処理に適用されることができる。

【００１８】

【発明の効果】本発明によれば西暦１９００年台で使用した既存のファイルやアプリケーションを変更することなく西暦２０００年台まで連続して運用することが可能となり，切替えに必要な保守費用を大幅に削減することができる。

【図面の簡単な説明】

【図１】本発明の原理説明図である。

【図２】本発明実施例によるデータ処理システムの構成図である。

【図３】従来例の年月日キー管理方法の説明図である。

【符号の説明】

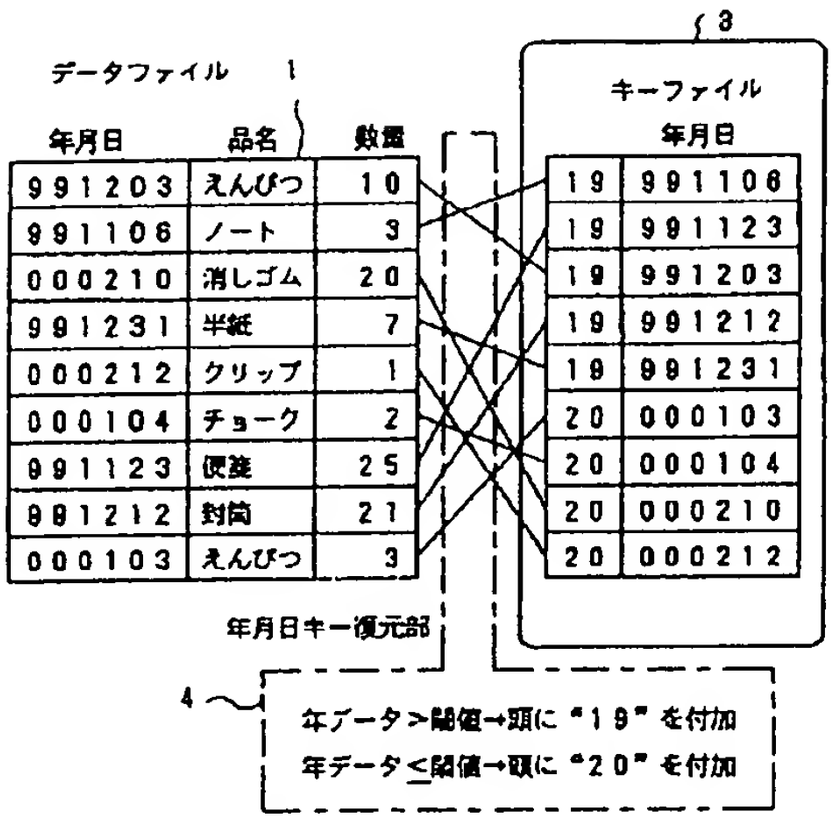
１　データファイル

３　キーファイル

４　年月日キー復元部

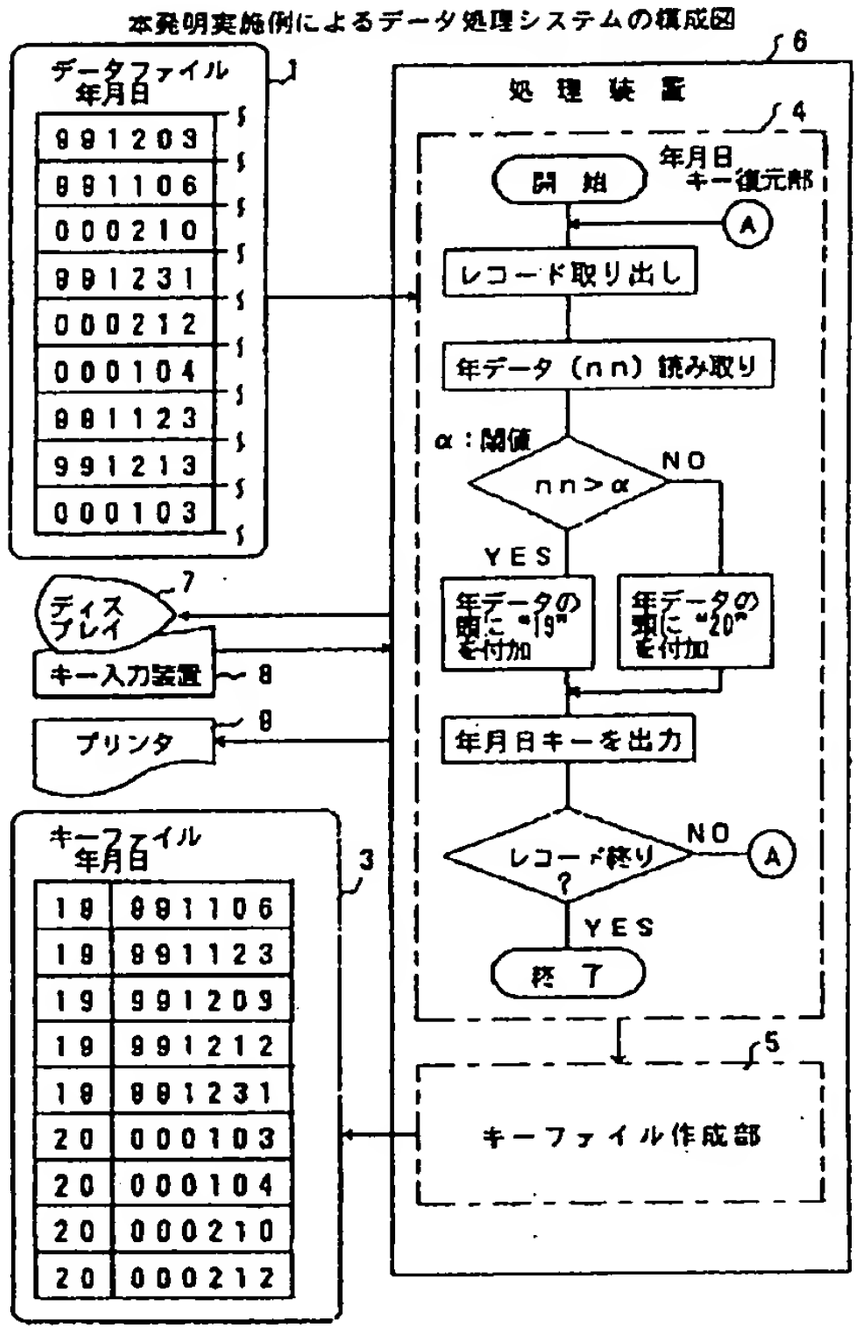
【図１】

本発明の原理説明図



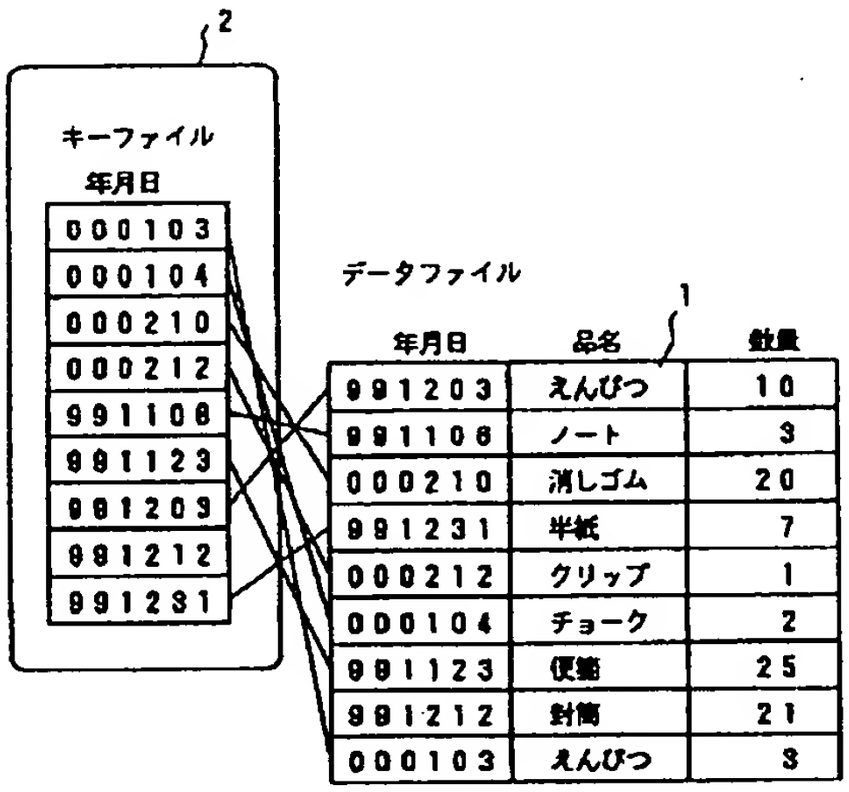
【図２】

本発明実施例によるデータ処理システムの構成図



【図３】

年月日キー管理方法の従来例の説明図



TRANSLATION FROM JAPANESE

(19) Japanese Patent Office (JP)	(12) Laid-open Patent Publication (A)	(11) Laid-open patent publication number Laid-open H. 6-103133

(43) Date of laying-open: 15th April 1994

(51) Int. Cl.[5]	Identifying symbols	Internal Patent Office Filing Number	FI	Location for indicating technical field
G06F 12/00	520 A	8526-5B		

Request for examination: not yet filed; Number of claims: 1; (4 pages in all)

(21) Application number	Patent Application H. 4-249902	(71) Applicants	000136136 BFU Co. Ltd. Aza Unoki 98-2, Unoki-cho, Kahoku-gun, Ishikawa-ken
(22) Application date	18th September 1992	(72) Inventor	Nobuo Saka c/o BFU Co. Ltd. Aza Unoki 98-2, Unoki-cho, Kahoku-gun, Ishikawa-ken
		(74) Agent	Patent attorney Fumihiro Hasegawa (and 2 others)

(54) [Title of the Invention] Method of managing date keys of a data file

(57) [Abstract]

[Object] This relates to a method of managing date keys in a data processing system in which a data file is processed having a date key of abbreviated type omitting the two most significant digits of the western calendar year; in particular, its object is to provide a date key management method which is useful in processing a data file in which, when moving from the 20th century to the 21st century of the western calendar year, dates of both centuries are intermingled.

[Constitution] It has an arrangement wherein, when compiling a key file of date keys for a data file or when performing sorting of data using a date key, the year data in the date key is compared with a prescribed threshold value and, depending on the result, the 2-digit value 19 or 20 is appended at the head of the year data to restore the year data to the 4-digit western calendar year, and key file compilation or data sorting is performed by means of a date key including this restored year data.

Diagram of the principles of the invention

Data file 1

Date	Product name	Nunerical quantity
991203	Pencil	10
991106	Notepad	3
000210	Eraser	20
991231	Rice paper	7
000212	Clipboard	1
000104	Chalk	2
991123	Writing paper	25
991212	Envelope	21
000103	Pencil	3

Key file 3

	Date
19	991106
19	991123
19	991203
19	991212
19	991231
20	000103
20	000104
20	000210
20	000212

Date key restoration unit 4

Year data > threshold value → append "19" at the head

Year data ≤ threshold value → append "20" at the head

[Claims]

[Claim 1] Method of managing date keys of a data file characterised in that in a data processing system comprising a data file in which only the least significant two digits, omitting the most significant two digits of the western calendar date are employed for the year data of the date key, when compiling a key file of date keys for said data file or when performing sorting of data using a date key, the year data in the date key is compared with a prescribed threshold value and, depending on the result,

the 2-digit value 19 or 20 is appended at the head of the year data to restore the year data to the 4-digit western calendar year, and key file compilation or data sorting is performed by means of a date key including this restored year data.

[Detailed description of the invention]

[0001]

[Field of Industrial Application] The invention relates to a method of managing a date key in a data processing system in which a data file is processed having an abbreviated date key in which the most significant two digits of the year in the western calendar are omitted. In particular, it provides a method of managing a date key which is useful for processing a data file in which, when the year in the western calendar shifts from the 20th century to the 21st century, date keys of both centuries are intermingled.

[0002]

[Prior art] Usually, when managing date data in a file, an abbreviated form is employed in which the most significant two digits of the year in the western calendar are omitted. This is because, at the present time, the value of the most significant two digits of the year in the western calendar is always "19", so the data is redundant. However, when the situation is reached in which data beyond the year 2000 in the western calendar are to be managed, the year data in abbreviated form becomes "00", so, when performing processing in which the date is used as key, inconvenience occurs in the order of arrangement of the data.

[0003] A prior art example is shown in Figure 3. In this Figure, 1 is a product management data file, in which each record comprises date data indicating date of purchase, product name data, and numerical quantity data. In the date data, for example "991203" represents 3rd December, 1999, and "000210" represents 10th February 2000. The arrangement of records within this data file 1 is random with respect to the value of the date data, which constitutes the key. 2 is a key file compiled for the date data of the records of data file 1, in which the date data are arranged in accordance with the magnitude of their value.

[0004] In the key file 2 of this prior art example, keys in which the value of the most significant two digits of the date data is "00" are arranged prior to the keys in which this is "99", so, at the boundary, an inversion in time sequence occurs, which is inconvenient. In general, this problem occurs when date data are sorted.

[0005]

[Problem that the invention is intended to solve] An object of this invention, in a data file in which date data providing a key are constituted using the least significant two digits of the year of the western calendar, is to make possible the compilation of a key file in which there is no inversion of time sequence and no need for alteration of

the data file even if the year 2000 in the western calendar is exceeded.

[0006]

[Means for solving the problem] In a method of managing date keys according to the invention, when compiling a key file or performing sorting keyed by date in a data file in which the least significant two digits of the western calendar year are employed as the year data, it is ascertained whether the value of the two digits of the year data belongs to the 20th century or 21st century of the western calendar and, in accordance with the result, the 2-digit year data is expanded by appending at its head the 2-digit value "19" or "20", thereby restoring it to the 4-digit western calendar year, and compilation of a key file or sorting is performed using the year data including this restored western calendar year.

[0007] The principle of the method of managing a date key according to the invention is that, for 2-digit year data of a data file, normally the range of values of the 20th century of the western calendar and the range of values of the 21st century of the western calendar are clearly distinguished, so by employing a suitable threshold value, the respective value ranges can easily be separated.

[0008] Figure 1 is a diagram of the principles of the invention. In this Figure, 1 is a data file whose record data have a date key including as year data the least significant two digits of the year of the western calendar.

[0009] 3 is a key file which is compiled using a date key including the 4-digit western calendar year data restored by date key restoration unit 4. 4 is the date key restoration unit; this compares the 2-digit value of the year data with a prescribed threshold value and, if the year data is greater than the threshold value, appends "19" at the head thereof, and, if the year data is smaller than the threshold value, appends "20" at the head thereof, thereby restoring a date key including 4-digit western calendar year data.

[0010]

[Action] The action of this invention will be described using Figure 1. In the date data (keys) of each of the records shown by way of example in the data file 1 of Figure 1, the two head digits are "99" or "00". This indicates that the corresponding year of the western calendar is 1999 or 2000. The reason for this is that the data file 1 does not contain the year data "2099" or "1900". In this case, by employing an arbitrary value between "99" and "00" as the threshold value, and comparing this with the year data in the date keys, it is simple to identify whether a date key belongs to the year 1999 or the year 2000. For example, if the threshold value is set to "50", if the year data is "99", which is larger than "50", "19" is appended to the head thereof, or, if the year data is "00", which is smaller than "50","20" is appended to the head thereof.

[0011] In general, if the minimum value of the year data in the 20th century is

"$n_0\,n_1$", and the maximum value of the year data in the 21st century is "$n_2\,n_3$", for the threshold value, a suitable value between "$n_0\,n_1$" and "$n_2\,n_3$" is employed.

[0012] Since the values of year keys having 4-digit western calendar year data which are restored in this way now accurately reflect the time sequence, the date keys of a key file 3 compiled by sorting these are put in correct time sequence order as shown in the drawing.

[0013]

[Embodiments] Figure 2 shows the layout of a data processing system according to an embodiment of the invention. In Figure 2, data file 1, key file 3, and date key restoration unit 4 are respectively the same as those described with reference to Figure 1. Also, 5 is a key file compilation unit that compiles a key file 3 from the date keys restored by date key restoration unit 4. 6, 7, 8, and 9 indicate the hardware construction of the system, 6 being a processing device, 7 being a display, 8 being a key input device and 9 being a pripter.

[0014] The function of date restoration unit 4 and key file compilation unit 5 is realised by a program. In the example illustrated in the drawings, a layout is adopted whereby key file compilation unit 5 is cascade-connected to date key restoration unit 4, and key file 3 is compiled by receiving the date key restored by reading from data file 1 by date key restoration unit 4; however, a layout could be adopted in which key file compilation unit 5 reads the date key from data file 1 and relies on restoration processing performed by calling date key restoration unit 4, or a layout could be adopted in which key file compilation unit 5 incorporates the function of date key restoration unit 4 internally. Further, a program having the function of the date key restoration unit could be provided by the system as a support program, or could be prepared as an application program.

[0015] In the layout of Figure 2, the date key restoration unit 4 operates as follows. First of all, it gets one record from data file 1, and reads the 2-digit year data (n n) in the date key thereof. Then, it compares the year data (n n) with the previously set threshold value (α) and, if n n > α, it appends "19" to the head of the year data (n n); otherwise, it appends "20". In this way, it restores the 4-digit year data, and, combining this with the remaining month and day data, transfers it to key file compilation unit 5. This processing is executed for the successive records of data file 1 and terminates when the last record is processed.

[0016] Key file compilation unit 5 writes into a working region of memory, not shown, all the date keys that have been obtained from date key restoration unit 4 and compiles a key file 3 arranged for example in ascending order by performing sorting processing, eliminating duplicated keys.

[0017] Although this invention has been described taking as example the

compilation of a key file, it is not restricted to this and can be applied to any desired file processing involving sorting of a date key.

[0018]

[Benefit of the Invention] With this invention, it becomes possible to use existing files or applications used in the 20th century of the western calendar into the 21st century of the western calendar continuously without having to modify them, thereby making it possible to greatly reduce maintenance costs necessary for changeover.

[Brief description of the drawings]

[Figure 1] This is a diagram of the principles of the invention.

[Figure 2] This is a layout diagram of a data processing system according to an embodiment of the invention.

[Figure 3] This is a diagram of a method of date key management according to a prior art example.

[Explanation of the reference symbols]

1 data file

3 key file

4 date key restoration unit

Drawings

Figure 1

Diagram of the principles of the invention

Data file 1

Date	Product name	Numerical quantity
991203	Pencil	10
991106	Notepad	3
000210	Eraser	20
991231	Rice paper	7
000212	Clipboard	1
000104	Chalk	2
991123	Writing paper	25
991212	Envelope	21
000103	Pencil	3

Key file 3

	Date
19	991106
19	991123
19	991203
19	991212
19	991231
20	000103
20	000104
20	000210
20	000212

Date key restoration unit 4

Year data > threshold value → append "19" at the head

Year data ≤ threshold value → append "20" at the head

Figure 2

Layout of data processing system according to embodiment of the invention

1 Data file date

7 Display

8 Key input device

9 Printer

3 Key file date

6 Processing device

4 Date key restoration unit

Start

Get record

Read year data (n n)

α: threshold value

YES

append "19" at the head of the year data

NO

append "20" at the head of the year data

Output date key

Record ends?

End

5 Key file compilation unit

Figure 3

Diagram of prior art example of date key management system

2 key file

Date

Data file 1

Date	Product name	Numerical quantity
991203	Pencil	10
991106	Notepad	3
000210	Eraser	20
991231	Rice paper	7
000212	Clipboard	1
000104	Chalk	2
991123	Writing paper	25
991212	Envelope	21
000103	Pencil	3

08/725574

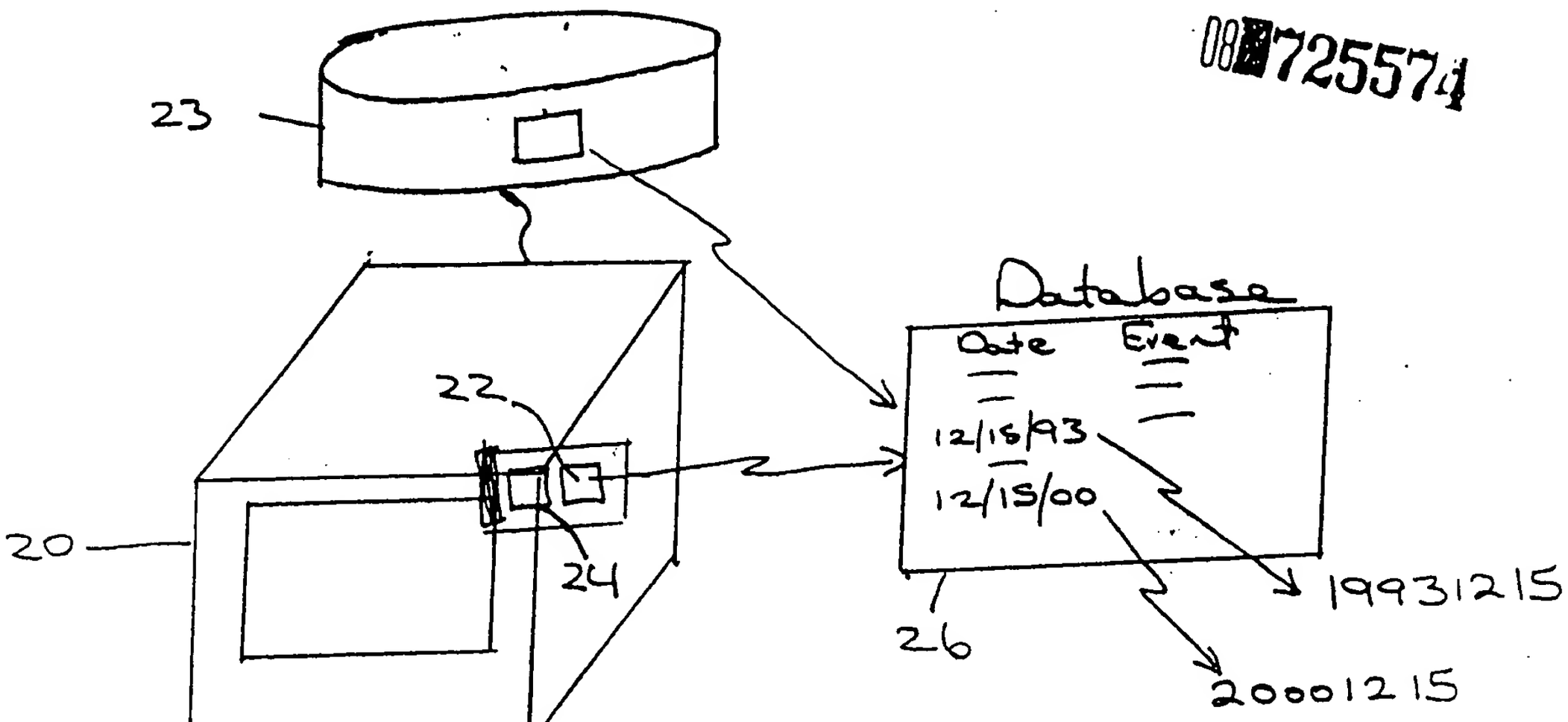
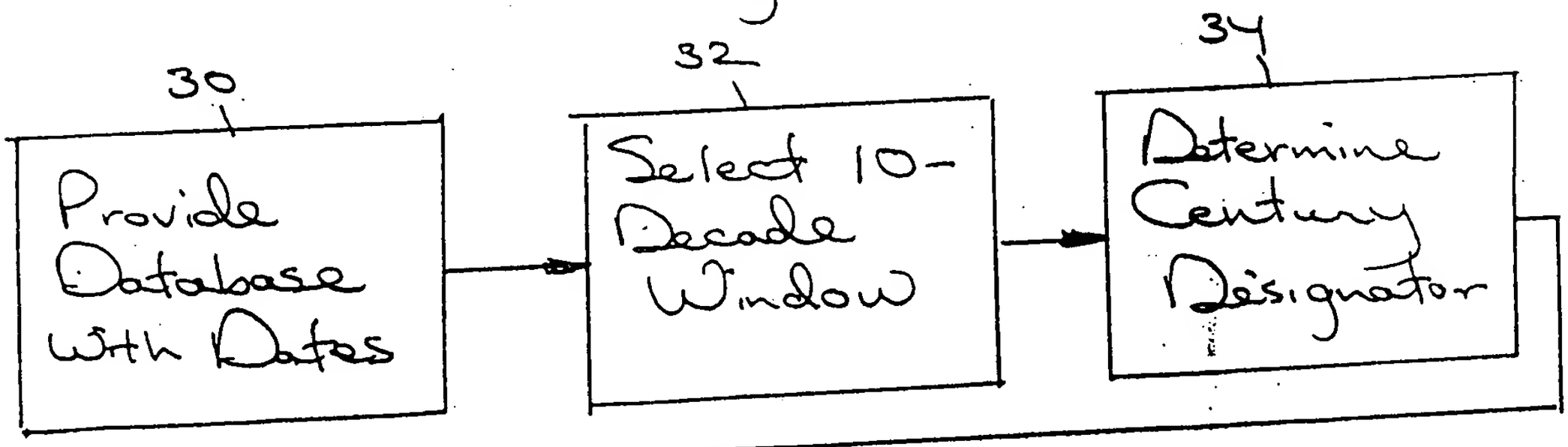


Fig 1



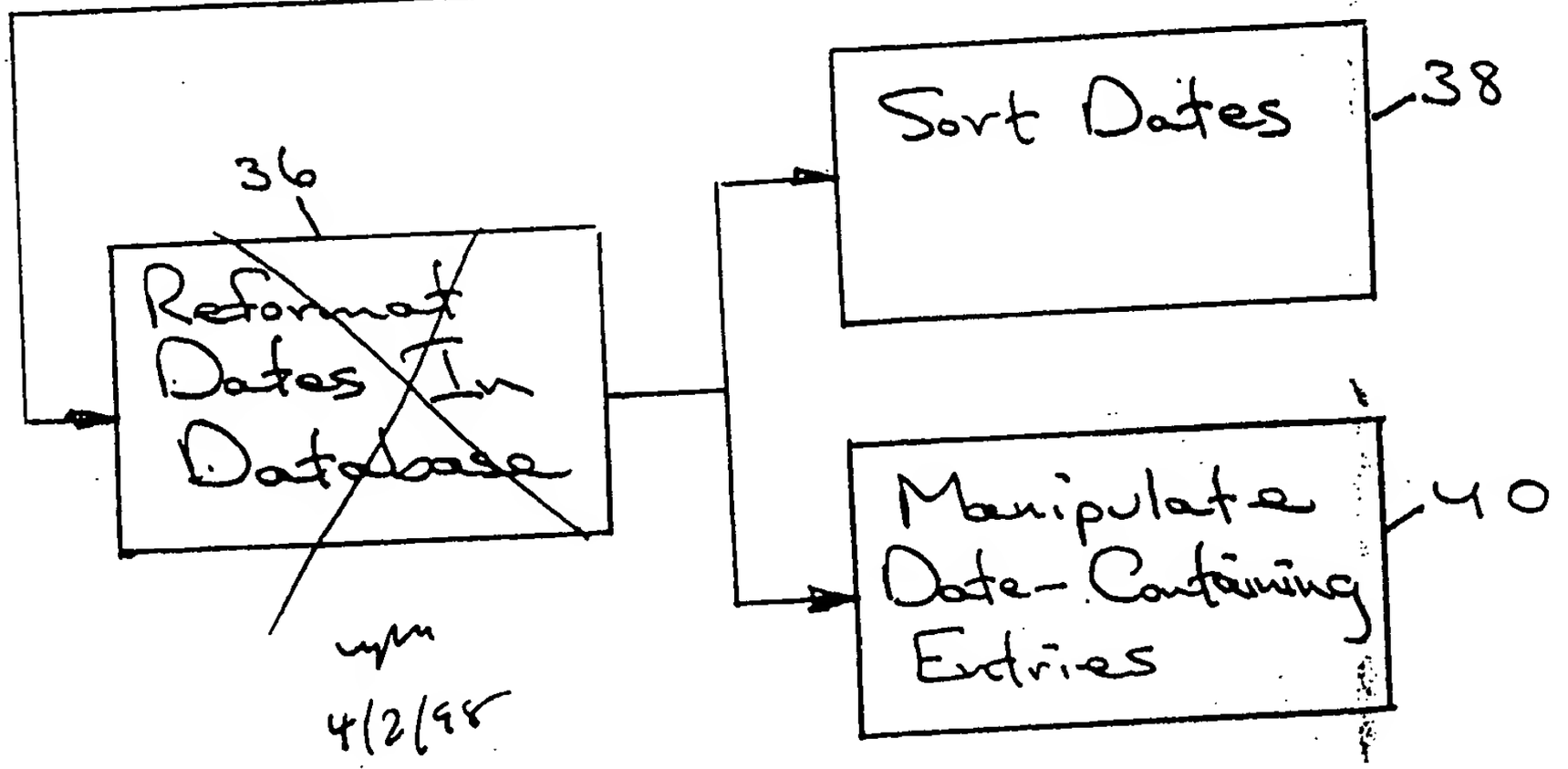


Fig 2

1.(with amendments) A method of processing symbolic representations of dates stored in a database, comprising the steps of

providing a database with symbolic representations of dates stored therein according to a format wherein M_1M_2 is the numerical month designator, D_1D_2 is the numerical day designator, and Y_1Y_2 is the numerical year designator, all of the symbolic representations of dates falling within a 10-decade period of time;

selecting a 10-decade window with a Y_AY_B value for the first decade of the window, Y_AY_B being no later than the earliest Y_1Y_2 year designator in the database;

determining a century designator C_1C_2 for each symbolic representation of a date in the database, C_1C_2 having a first value if Y_1Y_2 is less than Y_AY_B and having a second value if Y_1Y_2 is equal to or greater than Y_AY_B ; and

reformatting the symbolic representation of the date [in the database] with the values C_1C_2, Y_1Y_2, M_1M_2, D_1D_2 <u>to facilitate further processing of the dates.</u>

11.(with amendments) A method of processing dates in a database, comprising the steps of

providing a database with symbolic representations of dates stored therein according to a format wherein M_1M_2 is the numerical month designator, D_1D_2 is the numerical day designator, and Y_1Y_2 is the numerical year designator, all of the symbolic representations of dates falling within a 10-decade period of time which includes the decade beginning in the year 2000;

selecting a 10-decade window with a Y_AY_B value for the first decade of the window, Y_AY_B being no later than the earliest Y_1Y_2 year designator in the database;

determining a century designator C_1C_2 for each date in the database, C_1C_2 having a first value if Y_1Y_2 is less than Y_AY_B and having a second value if Y_1Y_2 is equal to or greater than Y_AY_B ; and

reformatting each date [in the database] <u>in the form $C_1C_2Y_1Y_2M_1M_2D_1D_2$ to facilitate further processing of the dates</u> and

sorting the dates [in the database] <u>in the form $C_1C_2Y_1Y_2M_1M_2D_1D_2$</u> [using a numerical-order sort].

```
!                -- Century Conversion --
!
!                Bruce Dickens   Apr 04, 1996
!
10    open structure tools:name 'otms_src_dir:tools'
      open #2 : name 'last_inv.dat', access output
         print "    Tools 'Last Inventory Data Format' Check for 1996 Inventory"
         print "ToolNo                ";" Model No  ";" LAST_INV ";"LAST_INV "
         print "======                ";" ========  ";" ======== ";"======== "
         print "Extract Data:"
         print #2: "ToolNo                ";" Model No  ";" LAST_INV ";"LAST_I
NV "
         print #2: "======                ";" ========  ";" ======== ";"======
== "
         print #2: "Extract Data:"

20    extract structure tools
         yy$ = lpad$ (element$(tools(last_inv),3,"/"), 2, "0" )
         mm$ = lpad$ (element$(tools(last_inv),1,"/"), 2, "0" )
         dd$ = lpad$ (element$(tools(last_inv),2,"/"), 2, "0" )
         cc$= yy$ + "/" +  mm$ + "/" + dd$
         c1$ = change$(cc$,'/','')
         if c1$[1:2] < '50' then
           c$ = '20' + c1$
         else
           c$= '19' + c1$
         end if
!        include c$ < '19960101'
           sort by tools(model)
           sort by rpad$(c$,8, '0')
!       if c$[1:8] < '19960101'  then
         print tools(toolno); tab(23); tools(model);  &
                        tab(35);tools(last_inv); tab(44); c$
         print #2: tools(toolno); tab(23); tools(model);  &
                        tab(35);tools(last_inv); tab(44); c$
                if valid ( c1$, "digits" ) = 0 then
                print; tab(53); " Date format is not digits"
                print #2:; tab(53); " Date format is not digits"
                end if
!               if valid ( c1$, "minlength 6" ) = 0 then
!               print ;tab(50); " Date format is short"
!               print #2: ;tab(50); " Date format is short"
!               end if
                if tools(last_inv) = "" then
                print ;tab(53); " Date format is blank "
                print #2: ;tab(53); " Date format is blank "
                end if
!       end if
30    end extract
         print
         print "Sorted Data:"
         print
40    for each tools
         c1$ = change$(tools(last_inv),'/','')
         print tools(toolno); tab(23); tools(model); &
          tab(35); tools(last_inv); tab(44); c$
         print #2: tools(toolno); tab(23); tools(model); &
          tab(35); tools(last_inv); tab(44); c$
                if valid ( c1$, "digits" ) = 0 then
                print ;tab(53); " Date format is not digits"
                print #2: ;tab(53); " Date format is not digits"
                end if
!               if valid ( c1$, "minlength 6" ) = 0 then
                print ;tab(53); " Date format is short"
                print #2: ;tab(53); " Date format is short"
                end if
```

Exhibit A

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 5,806,063

DATED : September 8, 1998

INVENTOR(S) : Dickens

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On title page, item [56],

In the References Cited, OTHER PUBLICATIONS, line 1, before "The", insert --IBM: --.

In the ABSTRACT, line 4, "$_{M1}M_2$" should read --M_1M_2--.

Signed and Sealed this

Twenty-ninth Day of December, 1998

Attest:

BRUCE LEHMAN

Attesting Officer

Commissioner of Patents and Trademarks

Patent 5,806,063, Reissue application, Claim 1	Japan 06-103133, April 15, 1994 [Citations are to the paragraph numbers in the text of both the Japanese publication and in the translation]
A method of processing symbolic representations of dates stored in a database, comprising the steps of	The reference is directed at managing date keys of a data file,[Title] which is effected by processing date representations in a database, each item of data is a symbolic representation.
providing a database with symbolic representations of dates stored therein according to a format wherein M_1M_2 is the numerical month designator, D_1D_2 is the numerical day designator, and Y_1Y_2 is the numerical year designator, all of the symbolic representations of dates falling within a 10-decade period of time;	The unprocessed database uses two digits to represent year data, see the data in date file 1, an example is the first entry, "991203" which represents 3rd Dec. 1999 [0003]. The text [0010 and 0011] make it clear that the date range is limited. "The reason for this is that the data file 1 does not contain the year data '2099' or '1900'.", there is a "minimum value of the year data in the 20th century" and a "maximum value of the year data in the 21st century" with the "threshold value" in between these two. This is only possible if the span of the data base is less than 10 decades.
selecting a 10-decade window with a Y_AY_B value for the first decade of the window, Y_AY_B being no later than the earliest Y_1Y_2 year designator in the database;	The "threshold value" or α corresponds to Y_AY_B and it is "no later" than the earliest Y_1Y_2 since it is selected as between n_0n_1, the minimum value of the 20th Century date range and the lower value, n_2n_3, which is the maximum value of the 21st Century date range [0011].
determining a century designator C_1C_2 for each symbolic representation of a date in the database, C_1C_2 having a first value if Y_1Y_2 is less than Y_AY_B and having a second value if Y_1Y_2 is equal to or greater than Y_AY_B, and	A comparison is made between the date data, nn, and the threshold value, α; if nn > α, the century designator "19" is used, otherwise, that is if nn $\leq \alpha$, the other century designator, "20" is used [0015]. Note also that the processing is applied to "the successive records of data file 1 and terminates when the last record is processed", i.e., the processing is applied to "each" record. [0015]
reformatting the symbolic representation of the date with the values C_1C_2, Y_1Y_2, M_1M_2, and D_1D_2 to facilitate further processing of the dates.	The date data, augmented with the century designator (19 or 20), is then written to key file 3; as seen there the date data has been reformatted to add the century designator.

Reissue application	Japan 06-103133, April 15, 1994 [Citations are to the paragraph numbers in the text of both the Japanese publication and in the translation]
2. The method of claim 1, wherein the 10-decade window includes the decade beginning in the year 2000.	The reference is directed to Y2K, i.e., the transition from the 20th to the 21st century and so, by definition, uses a window which encompasses the year 2000 [Object].
3. The method of claim 2, wherein the step of determining includes the step of determining the first value as 20 and the second value as 19.	Since the reference is directed at Y2K [Object] the century indicators are "19" and "20" [Constitution].
4. The method of claim 1, including an additional step, after the step of reformatting, of sorting the symbolic representations of dates.	After 4 digit year value is determined, "data sorting" is performed [Constitution].
5. The method of claim 1, wherein the step of reformatting includes the step of reformatting each symbolic representation of a date into the format $C_1C_2Y_1Y_2M_1M_2D_1D_2$.	Once the proper century indicator is determined, it is "appended" to the year data so as to combine the 4 digit year with month and day data [0015], this is the claimed format.
6. The method of claim 5, including an additional step, after the step of reformatting, of sorting the symbolic representations of dates using a numerical-order sort.	Once the eight digit date data (four digit year, two digit month and day) is created, the key file 3 is compiled by "sorting" [0016]. A numerical sort can be used since, the eight digit date data "now accurately reflect the time sequence" [0012].

Reissue application	Japan 06-103133, April 15, 1994 [Citations are to the paragraph numbers in the text of both the Japanese publication and in the translation]
8. The method of claim 1, wherein the step of selecting includes the step of selecting Y_AY_B such that Y_B is 0 (zero).	The reference proposes a "threshold value" of 50 [Action] which corresponds to $Y_A = 5$ and $Y_B = 0$.
9. The method of claim 1, including an additional step, after the step of reformatting, of storing the symbolic representation of dates and their associated information back into the database.	The key file 3 has the restored date keys and it is part of the database [0016] to correspond to this clause.
10. The method of claim 9, including the additional step, after the step of reformatting, of manipulating information in the database having the reformatted date information therein.	The act of manipulating information is the purpose of any database - it is inherent in the reference.

Patent 5,806,063, Reissue application, claim 11	Japan 06-103133, April 15, 1994 [Citations are to the paragraph numbers in the text of both the Japanese publication and in the translation]
A method of processing dates in a database, comprising the steps of	The reference is directed at managing date keys of a data file,[Title] which is effected by processing date data in a database.
providing a database with dates stored therein according to a format wherein M_1M_2 is the numerical month designator, D_1D_2 is the numerical day designator, and Y_1Y_2 is the numerical year designator, all of dates falling within a 10-decade period of time which includes the decade beginning in the year 2000;	The unprocessed database uses two digits to represent year data, see the data in date file 1, an example is the first entry, "991203" which represents 3rd Dec. 1999 [0003]. The text [0010 and 0011] make it clear that the date range is limited "The reason for this is that the data file 1 does not contain the year data '2099' or '1900'.", there is a "minimum value of the year data in the 20th century" and a "maximum value of the year data in the 21st century" with the "threshold value" in between these two. This is only possible if the span of the data base is less than 10 decades. Finally it is apparent that the 10 decade period includes the decade beginning with the year 2000.
selecting a 10-decade window with a Y_AY_B value for the first decade of the window, Y_AY_B being no later than the earliest Y_1Y_2 year designator in the database;	The "threshold value" or α corresponds to Y_AY_B and it is "no later" than the earliest Y_1Y_2 since it is selected as between n_0n_1, the minimum value of the 20th Century date range and the lower value, n_2n_3, which is the maximum value of the 21st Century date range[0011].
determining a century designator C_1C_2 for each date in the database, C_1C_2 having a first value if Y_1Y_2 is less than Y_AY_B and having a second value if Y_1Y_2 is equal to or greater than Y_AY_B ;	A comparison is made between the date data, nn, and the threshold value, α; if $nn > \alpha$, the century designator "19" is used, otherwise, that is if $nn \leq \alpha$, the other century designator, "20" is used [0015]. Note also that the processing is applied to "the successive records of data file 1 and terminates when the last record is processed", i.e., the processing is applied to "each" record. [0015]
reformatting each date in the form $C_1C_2Y_1Y_2M_1M_2D_1D_2$ to facilitate further processing of the dates; and	The date data has the selected century designator appended. "In this way, it restores the 4-digit year data, and, combining this with the remaining month and day data, transfers it to the key file compilation unit 5". [0015] That is, we start with $Y_1Y_2M_1M_2D_1D_2$ and append C_1C_2, to end up with $C_1C_2Y_1Y_2M_1M_2D_1D_2$. Note also that

	the processing is applied to "the successive records of data file 1 and terminates when the last record is processed", i.e., the processing is applied to "each" record. [0015]
sorting the dates in the form $C_1C_2Y_1Y_2M_1M_2D_1D_2$	The key file compilation unit 5 arranges the data in ascending order "by performing sorting processing …". [0016]